A PROJECT REPORT ON
ANALYSIS ON FAKE JOBS POSTING

This project dissertation is submitted to Osmania University, as

Partial fulfilment for the Completion of M.Sc. (Statistics) IV-Semester

Sirisha Gani	1007-23-507-002
Akkenapalli Mandira	1007-23-507-014
Sanagonda Shylaja	1007-23-507-026
Sk Israk Sahan	1007-23-507-032
Ayesha Ruhi	1007-23-507-038

Under the supervision of

Prof. G. Jayasree

Department of Statistics
University College of Science
Osmania University, Hyderbad-500007

DECLARATION

We hereby declare that the project work presented in this dissertation, entitled **"A Project Report On Analysis on Fake jobs posting"** has been carried out by **us** under the supervision of **Prof. G. Jayasree , Head** , in the Department of Statistics, University College of Science, Osmania University, Hyderabad – 500 007. The work done is original and has not been submitted so far, in part or full, for any other degree or diploma to any University or institution.

1. Shirisha Gani :1007-23-507-002
2. Akkenapalli Mandira :1007-23-507-014
3. Sanagonda Shylaja :1007-23-507-026
4. Sk Israk Sahan :1007-23-507-032
5. Ayesha Ruhi :1007-23-507-038

Date:

Department of Statistics
University College of Science
Osmania University
Hyderabad-500007.

Date:

CERTIFICATE

This is to certify that the research work presented in this thesis, entitled **"A Project On Analysis and prediction on Fake Jobs Posting"** has been carried out by, registered M.Sc. (**Statistics**) IV-Semester students, of this University. The work done is original and has not been submitted so far, in part or full, for any other degree or diploma to any University or institution.

Date: Signature of the Supervisor

Signature of Head with office Seal

Date of Examination:

Internal Examiner Signature with date:

External Examiner Signature with date:

Acknowledgement

We are thankful to our supervisor **Prof. G. Jayasree, Head, Department of Statistics, University College of Science, Osmania University, Hyderabad - 500 007,** for her guidance.

We are also thankful to **Prof. G. Jayasree, Head Department of Statistics, University College of Science, Osmania University, Hyderabad-500 007** for providing necessary facilities for doing this project.

We would like to express our sincere thanks to our revered professors **Prof. N. Ch. Bhatra Charyulu, CBOS, Prof. S.A. Jyothi Rani, DR. G. Sirisha, DR. M. Raghavender Sharma, and Mrs. J.L. Padmashree, Department of Statistics, Osmania University** for their encouragement throughout the period of our course work.

We also extend our heartfelt gratitude to **Mrs. M. Sunitha Ma'am** for her unwavering support and guidance, which has been instrumental in our project work.

We extend our thanks to all non-teaching staff, Department of Statistics, for their secretarial help during the period.

1. Shirisha Gani :1007-23-507-002
2. Akkenapalli Mandira :1007-23-507-014
3. Sanagonda Shylaja :1007-23-507-026
4. Sk Israk Sahan :1007-23-507-032
5. Ayesha Ruhi :1007-23-507-038

CHAPTER 1

INTRODUCTION

CHAPTER-1
INTRODUCTION

1.1 Introduction

In today's digital age, online job portals and social media platforms have made job hunting easier and faster. However, they have also given rise to a serious problem — **fake job postings**. These are fraudulent job advertisements created by individuals or organizations with the intent to deceive job seekers. The motives behind such postings can vary, including collecting personal information, demanding money for fake recruitment processes, or scamming people with false promises of high-paying jobs.

Fake job postings not only waste a candidate's time and effort but can also lead to financial loss and identity theft. They often appear on legitimate job portals, emails, or social media, making them difficult to spot. Therefore, it is crucial for job seekers to verify the authenticity of job offers and be aware of common signs of fraud, such as requests for upfront payment, vague job descriptions, or suspicious contact information.

Fake job postings often target fresh graduates, unemployed individuals, and those urgently seeking work, as they are more likely to respond quickly to opportunities without verifying their legitimacy. Scammers exploit this vulnerability by offering attractive salaries, work-from-home options, or roles in reputed companies. Many fake recruiters also use social engineering tactics, such as posing as HR representatives, to gain the trust of applicants and extract sensitive information like bank details, identification numbers, or even money under the guise of "registration" or "training fees."

To tackle this issue, it is essential for job seekers to develop awareness and verify every opportunity before proceeding. Checking the official company website, avoiding offers that demand money upfront, and scrutinizing job descriptions for inconsistencies are basic steps to stay safe. Moreover, governments, educational institutions, and online platforms should take initiatives to educate people about online job scams. By staying alert and informed, job seekers can protect themselves and contribute to reducing the impact of fake job postings in society.

1.2. Data Description

The dataset is extracted from the source https://github.com and has been utilised for the predictions of Fake Jobs Posting using different methods of Natural Language Processing and Machine Learning.

Usage: Fake Jobs Posting
Format: A data frame with 17,886 observations following with the 18 variables.

We used the features like,

- Job_id : Unique ID for the job posting
- Title : Job title (e.g., “Software Engineer”)
- Location: Location (e.g., “US, NY, New York”).
- Department : Department name (e.g., “Sales”, “IT”)
- Salary_Ranges : Sales range offered
- Company_Profile : Overview or description of the company
- Description : Job description
- Requirement : Required skills and qualifications
- Benefits : Offered job benefits
- Telecommunicating : 1=remote allowed, 0=on-site
- Has_company_logo : 1=logo present, 0=absent
- Has_questions : 1=has screening questions
- Employment_type : Full-time, Part-time, Contract, etc.
- Required_experience : Experienced level required
- Required_education : Education qualification required
- Industry : Industry type(e.g., IT, Health Care)
- Function : Job function (e.g., Marketing, Sales)
- Fradualent : 1=fake job post, 0=genuine

1.3. Sample data

The following is the sample data set taken from the data frame of 17,886 observations using the "head()".

The data sample below consists of the first five observations of the dataframe which is being studied.

job_id	title	…	company_profile	description	requirements	…	fraudulent
1	Marketing Intern	…	We're Food52, …located in Chelsea, in New York City.	Food52, …executive staff	Experience with …working long hours	…	0
2	Customer Service - Cloud Video Production	…	90 Seconds, …7bff2ea17e#Â	Organised - Focused - Vibrant - … GridAKL!Â	What we … before reaching out.	…	0
3	Commissioning Machinery Assistant (CMA)	…	Valor Services …Your Success is Our Mission.Â â„¢Â	Our client, … safety regulations.	Implement pre-commissioning …projects internationally.	…	0
4	Account Executive - Washington DC	…	Our … professional growth, and much more.	THE COMPANY … to resolve issues	EDUCATION …, realistic, time-driven goals with clear lead indicators	…	0
5	Bill Review Manager	…	SpotSource … 60cf428551#	JOB TITLE: Itemization Review …customersÂ	QUALIFICATIONS: RN license …in Microsoft Office Suite	…	0

1.4. Objectives of the project

This project aims to design and evaluate a predictive model capable of detecting fake job postings using a combination of NLP techniques and advanced ML algorithms.

- Explore linguistic patterns and structural anomalies common in fake postings,
- Extract meaningful features using text processing and statistical analysis,
- Train multiple classification models including traditional ML algorithms,
- Compare model performances and interpret the most influential factors in determining the legitimacy of a job posting.

Scope: Ultimately, this research contributes to the broader effort of improving trust and security in online job markets. The findings have practical implications for job portals, and HR systems aiming to protect users from employment scams.

CHAPTER 2
DATA PRE-PROCESSING THROUGH NATURAL LANGUAGE PROCESSING

CHAPTER-2
DATA PRE-PROCESSING THROUGH NATURAL LANGUAGE PROCESSING

2.1 Introduction

Natural Language Processing (NLP) is a crucial field of artificial intelligence (AI) that focuses on the interaction between humans and computers using natural language. It combines computer science, linguistics, and machine learning to help computers understand, interpret, and generate human languages. In essence, NLP allows machines to process written and spoken words in a way that is both meaningful and useful. With the growing amount of digital text and speech data, NLP has become a powerful tool in various industries, revolutionizing the way we communicate with machines.

The working of NLP involves multiple steps and techniques. First, text pre-processing is carried out, where unnecessary characters, punctuation, and irrelevant data are removed. Then comes **tokenization**, where sentences are broken down into words or smaller parts. Next, techniques like **stemming** and **lemmatization** are applied to reduce words to their root forms. More advanced techniques involve **part-of-speech tagging** to identify nouns, verbs, adjectives, and more. After this, machine learning and deep learning models help computers understand the context and meaning of the text. This allows systems to not only process information but also generate meaningful and human-like responses.

Why is NLP Important?

- Humans use language to communicate, but computers use numbers. NLP acts as a bridge between human communication and machine understanding.
- It powers many tools we use daily, like Chabot's, search engines, virtual assistants (Alexa, Siri), translation apps, and spam filters**.**

2.2 Text Pre-Processing:

Text pre-processing is the first and most essential step in Natural Language Processing (NLP), where raw text is cleaned and transformed into a standardized form suitable for further analysis. It prepares unstructured data, such as job descriptions or social media posts, by removing noise and inconsistencies that could affect model performance. The following steps used in this dataset are,

1. Lowercasing

Lowercasing is one of the simplest but most important text preprocessing techniques. It converts all characters in the text to lowercase. This helps to ensure that words like "Job," "job," and "JOB" are treated as the same word instead of different tokens. For instance, in job posting analysis, both "Manager" and "manager" will be understood as the same term.

2. Removing URLs, Hashtags, Mentions, and Special Characters

Real-world text often contains noise such as hyperlinks, hashtags, user mentions, and symbols that are irrelevant for most NLP tasks. Removing these elements cleans the data by eliminating components like "http://jobs.com," "#hiring," or "@company," which do not add meaning to the context. This is especially useful in social media data, job descriptions, or web-scraped text where such elements are common.

3. Removing Punctuation

Punctuation marks like commas, periods, question marks, and exclamation points may not contribute to the analysis for tasks such as classification or clustering. By removing them, the text becomes simpler and reduces noise in the dataset. For example, "Apply now! Hurry…" becomes "Apply now Hurry." However, in some advanced NLP tasks like sentiment analysis, punctuation can carry meaning, so this step is applied only when appropriate.

4. Removing Stop Words

Stop words are common words such as "the," "is," "at," and "on," which occur frequently in text but carry little analytical value. Removing these words reduces the size of the dataset and helps models focus on more meaningful terms. For instance, the sentence "The company is hiring developers" becomes "company hiring developers," making it easier for algorithms to identify the key message.

5. Tokenization

Tokenization breaks text into smaller units such as words, subwords, or sentences. This is crucial because NLP models process text in pieces rather than as continuous strings. For example, the sentence "We are hiring Python developers" can be tokenized into ["We," "are," "hiring," "Python," "developers"].

6. Lemmatization

Lemmatization reduces words to their base or dictionary form (lemma) using linguistic rules. Unlike stemming, which simply chops off word endings, lemmatization produces meaningful root words. For example, "running," "ran," and "runs" all become "run." This ensures that different word forms are treated as the same term, which improves the model's understanding of context.

2.3 Feature extraction

Feature extraction in NLP is the process of converting text into numerical representations that machine learning models can process. Common methods include Bag-of-Words **(BoW)** and TF-IDF. More advanced techniques like word embedding (Word2Vec, GloVe) and contextual embedding (BERT) capture deeper semantic relationships between words. This step is essential for transforming unstructured text into features that enable effective text classification, clustering, and other NLP tasks.

1. Bag-of-Words (BoW):

The Bag-of-Words model represents text as a collection of its words, ignoring grammar and word order. It creates a **vocabulary** of all unique words in the dataset and counts how often each word appears in a document. For example, the sentences "I love NLP" and "I love Python" would generate a word matrix counting occurrences of "I," "love," "NLP," and "Python." BoW is simple and effective but does not capture the context or meaning of words.

2. TF-IDF (Term Frequency–Inverse Document Frequency):

TF-IDF improves on BoW by not only counting word frequency but also measuring the importance of words. It increases the weight of words that are frequent in one document but rare across the entire dataset, helping highlight key terms. For example, in job postings, common words like "job" may be down weighted, while unique words like "data scientist" get higher importance. This makes TF-IDF more useful for identifying significant keywords and improving model performance.

2.4 Applications of NLP

Natural Language Processing (NLP) has a wide range of applications in real-world scenarios where human language needs to be understood, processed, and acted upon by machines:

1. **Text Classification** – Categorizing texts into predefined labels (e.g., spam detection, fake job postings detection, sentiment classification).
2. **Sentiment Analysis** – Identifying the emotional tone (positive, negative, neutral) in reviews, social media posts, or feedback.
3. **Information Extraction** – Extracting key entities like names, dates, or locations from unstructured text (e.g., resumes, news).
4. **Machine Translation** – Translating text from one language to another (e.g., Google Translate).
5. **Chatbots & Virtual Assistants** – Building intelligent agents like Siri, Alexa, and customer support bots.
6. **Text Summarization** – Creating concise summaries from large documents or articles.

7. **Speech Recognition & Generation** – Converting speech to text and vice versa (e.g., voice assistants, transcription tools).
8. **Search Engines** – Improving search relevance using semantic understanding of queries.
9. **Fake Content Detection** – Detecting fraudulent job postings, fake reviews, or misleading news articles.

2.5 Future Scope

In the future, more advanced NLP techniques can be integrated to improve text understanding beyond basic TF-IDF features. Contextual word embedding like **BERT, RoBERTa, or XLNet** can replace traditional vectorization, allowing the model to capture deeper meaning and context in job postings. **Named Entity Recognition (NER)** could be used to extract structured information such as company names, job titles, and locations for more accurate analysis. **Topic modelling** techniques like **LDA (Latent Dirichlet Allocation)** could help identify hidden patterns or themes in postings. Additionally, incorporating **sentiment and intent analysis** could provide better insights into the tone and purpose of job descriptions. These NLP-focused improvements would make the system more intelligent, context-aware, and capable of handling complex textual data.

CHAPTER 2
DATA MODELLING THROUGH MACHINE LEARNING TECHNIQUES

CHAPTER-2
DATA MODELING THROGH MACHINE LEARNING TECHNIQUES

3.1 Introduction

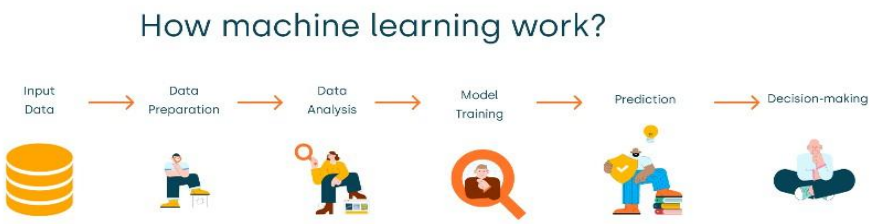
What is Machine Learning?

Machine learning (ML) is a branch of artificial intelligence (AI) focused on enabling computers and machines to imitate the way that humans learn, to perform tasks autonomously, and to improve their performance and accuracy through experience and exposure to more data.

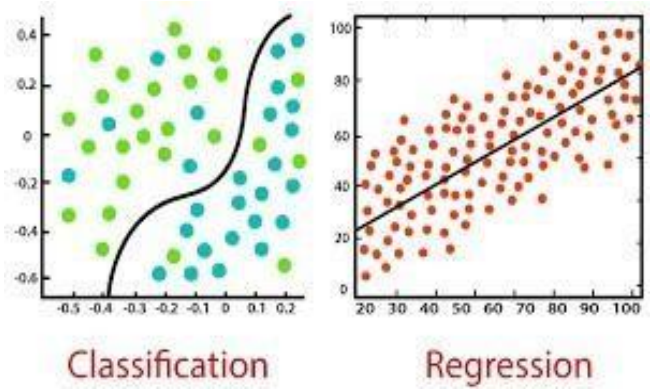
How Machine Learning Works?

Machine learning is a type of artificial intelligence (AI) that uses algorithms to learn from data and make predictions. Machine learning models can learn to identify patterns, recognize objects, and make decisions. Each step is important and cannot be skipped to achieve high accuracy. Code Implementation follows the below-mentioned steps:

- Data Collection
- Data Pre-processing
- Model Training
- Model Evaluation
- Model Deployment



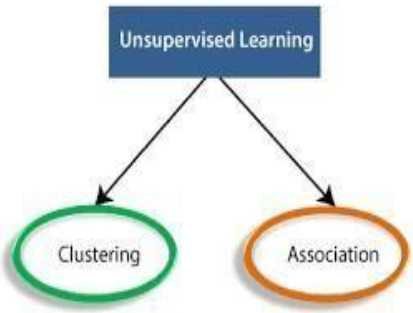
1. Supervised Learning: Supervised learning is a type of machine learning where the algorithm learns from labelled data, meaning each training example consists of input features along with corresponding output labels. The algorithm learns to map input features to the output labels by finding patterns and relationships in the data. This allows it to make predictions or decisions on new, unseen data based on the learned patterns.



Supervised learning tasks include classification, where the algorithm predicts a discrete label or category, and regression, where it predicts a continuous value.

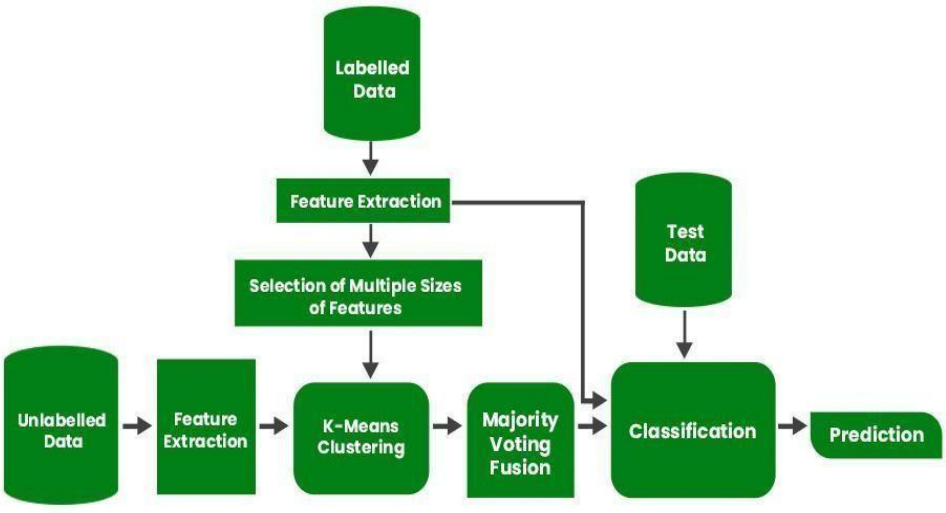
Regression Algorithm	Classification Algorithm
The output variable has to be a real value or continuous in nature.	The output variable is discrete in the Classification SML problem.
Regression algorithm helps to map the input value and the continuous output variable.	Classification algorithm helps in mapping the input value with the output variable which is discrete in nature.
Regression algorithms are only used for data that is continuous.	Classification algorithms are only used for data that is discrete.
Finds the best fit for accurately predicting the output data in Regression.	Finds the decision boundary which helps in classifying the dataset into different classes in Classification
Regression algorithms help in solving the problems like predicting the weather or the price of Houses in a neighbourhood according to a real estate trend.	Classification algorithms are useful for problems like identifying spam emails, recognition of speech, identifying cancer cells, etc.

2. Unsupervised Learning: Unsupervised learning is a type of machine learning where the algorithm learns from unlabelled data, meaning it doesn't have explicit output labels associated with the input features. Instead, the algorithm identifies patterns, structures, or relationships within the data without any guidance.



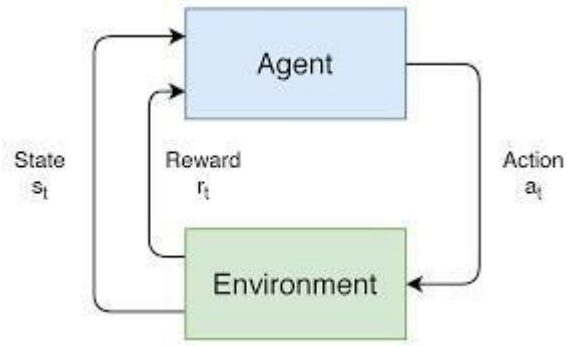
Unsupervised learning tasks include clustering, where the algorithm groups similar data points together, and dimensionality reduction, where the algorithm reduces the number of features while preserving important information.

3. Semi-supervised Learning: Semi-supervised learning is a type of machine learning that combines both labelled and unlabelled data for training. This approach is useful when labelled data is scarce or expensive to obtain. The algorithm learns from the small amount of labelled data available and utilizes the larger pool of unlabelled data to improve its performance.



Semi-supervised learning algorithms typically leverage the structure or distribution of the unlabelled data to generalize better to new, unseen examples. This approach can lead to improved model performance compared to using only labelled data, especially in scenarios where obtaining labelled data is challenging.

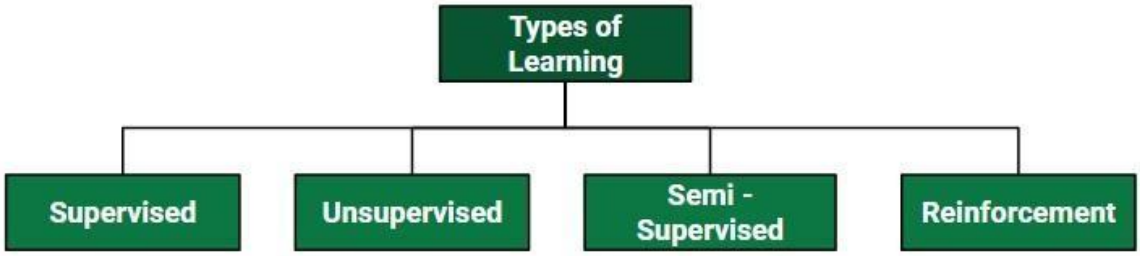
4. Reinforcement learning



Reinforcement learning is a type of machine learning where an agent learns to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. The goal is to maximize the total rewards accumulated over time.

3.2 Importance of Machine Learning

Machine learning is important because it gives enterprises a view of trends in customer behaviour and operational business patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google, and Uber, make machine learning a central part of their operations. Machine learning has become a significant competitive differentiator for many companies. Machine learning has several practical applications that drive the kind of real business results - such as time and money savings - that have the potential to dramatically impact the future of your organization. In particular, we see tremendous impact occurring within the customer care industry, whereby machine learning is allowing people to get things done more quickly and efficiently. Through Virtual Assistant solutions, machine learning automates tasks that would otherwise need to be performed by a live agent - such as changing a password or checking an account balance. This frees up valuable agent time that can be used to focus on the kind of customer care that humans perform best: high touch, complicated decision-making that is not as easily handled by a machine. At Interactions, we further improve the process by eliminating the decision of whether a request should be sent to a human or a machine: unique Adaptive Understanding technology, the machine learns to be aware of its limitations, and bailout to humans when it has low confidence in providing the correct solution.



Some important applications in which machine learning is widely used are given below 1. Healthcare: Machine Learning is widely used in the healthcare industry. It helps healthcare researchers to analyze data points and suggest outcomes. Natural language processing helped to give accurate insights for better results of patients. Further, machine learning has improved the treatment methods by analyzing external data on patients' conditions in terms of X-ray, Ultrasound, CT-scan, etc. NLP, medical imaging, and genetic information are key areas of machine learning that improve the diagnosis, detection, and prediction system in the healthcare sector.

2. Automation: This is one of the significant applications of machine learning that helps to make the system automated. It helps machines to perform repetitive tasks without human intervention. As a machine learning engineer and data scientist, you have the responsibilities to solve any given task multiple times with no errors. However, this is not practically possible for humans. Hence machine learning has developed various models to automate the process, having the capability of performing iterative tasks in lesser time.

3. Banking and Finance: Machine Learning is a subset of AI that uses statistical models to make accurate predictions. In the banking and finance sector, machine learning helped in many ways, such as fraud detection, portfolio management, risk management, chatbots, document analysis, highfrequency trading, mortgage underwriting, AML detection, anomaly detection, risk credit score detection, KYC processing, etc. Hence, machine learning is widely applied in the banking and finance sector to reduce error as well as time.

4. Transportation and Traffic Prediction: This is one of the most common applications of Machine Learning that is widely used by all individuals in their daily routine. It helps to ensure highly secured routes, generate accurate ETAs, predict vehicle breakdown, Driving Prescriptive Analytics, etc. Although machine learning has solved transportation problems, it still requires more improvement. Statistical machine learning algorithms helps to build a smart transportation system. Further, deep Learning explored the complex interactions of roads, highways, traffic, environmental elements, crashes, etc. Hence, machine learning technology has improved daily traffic management as well as a collection of traffic data to predict insights of routes and traffic.

5. Image Recognition: It is one of the most common applications of machine learning which is used to detect the image over the internet. Further, various social media sites such as Facebook uses image recognition for tagging the images to your Facebook friends with its feature named auto friend tagging suggestion. Further, now a day's, almost all mobile devices come with exciting face detection features. Using this feature, you can secure your mobile

data with face unlocking, so if anyone tries to access your mobile device, they cannot open without face recognition.

6. Speech Recognition: Speech recognition is one of the biggest achievements of machine learning applications. It enables users to search content without writing text or, in other words, 'search by voice'. It can search content/products on YouTube, Google, Amazon, etc. platforms by your voice. This technology is referred to as speech recognition. It is a process of converting voice instructions into the text; hence it is also known as 'Speech to text' or 'Computer speech recognition. Some important examples of speech recognitions are Google assistant, Siri, Cortana, Alexa, etc.

7. Product Recommendation: It is one of the biggest achievements made by machine learning which helps various e-commerce and entertainment companies like Flipkart, Amazon, Netflix, etc., to digitally advertise their products over the internet. When anyone searches for any product, they start getting an advertisement for the same product while internet surfing on the same browser. This is possible by machine learning algorithms that work on users' interests or past experience and accordingly recommend them for products. For e.g., when we search for a laptop on the Amazon platform, then it also gets started with so many other laptops having the same categories and criteria. Similarly, when we use Netflix, we find some recommendations for entertainment series, movies, etc. Hence, this is also possible by machine learning algorithms.

8. Virtual Personal Assistance: This feature helps us in many ways, such as searching content using voice instruction, calling a number using voice, searching contact in your mobile, playing music, opening an email, Scheduling an appointment, etc. Now a day, you all have seen advertising like "Alexa! Play the Music" this is also done with the help of machine learning. Google Assistant, Alexa, Cortana, Siri, etc., are a few common applications of machine learning. These virtual personal assistants record our voice instructions, send them over to the server on a cloud, decode it using ML algorithms and act accordingly.

9. Email Spam and Malware detection & Filtering: Machine learning also helps us for filtering emails in different categories such as spam, important, general, etc. In this way, users can easily identify whether the email is useful or spam. This is also possible by machine learning algorithms such as Multi-Layer Perceptron, Decision tree, and Naïve Bayes classifier. Content filter, header filter, rules-based filter, permission filter, general blacklist filter, etc., are some important spam filters used by Google.

10. Self-driving cars: This is one of the most exciting applications of machine learning. Machine learning plays a vital role in the manufacturing of self-driving cars. It uses an unsupervised learning method to train car models to detect people and objects while driving. Tata and Tesla are the most popular car manufacturing companies working on self-driving cars. Hence, it is a big revolution in a technological era which is also done with the help of machine learning.

11. Credit card fraud detection: Credit card frauds have become very easy targets for online hackers. As the culture of online/digital payments is increasing, the risk of credit/debit cards is parallel increasing. Machine Learning also helps developers to detect and analyze frauds in online transactions. It develops a novel fraud detection method for Streaming Transaction Data, with an objective to analyze the past transaction details of the customers and extract the behavioural patterns. Further, cardholders are clustered into various categories with their transaction amount so that the behavioural pattern of the groups can be extracted respectively.

Hence, credit card fraud detection is a novel approach using Aggregation Strategy and Feedback Mechanism of machine learning.

12. Stock Marketing and Trading: Machine learning also helps in the stock marketing and trading sector, where it uses historical trends or past experience for predicting the market risk. As share marketing is another name of marketing risk, machine learning reduces it to some extent and predicts data against marketing risk. Machine learning's long short-term neural memory network is used for the prediction of stock market trends.

13. Language Translation: The use of Machine learning can be seen in language translation. It uses the sequence-to-sequence learning algorithms for translating one language into other. Further, it also uses images recognition techniques to identify the text from one language to other. Similarly, Google's GNMT (Google Neural Machine Translation) provides this feature, which is a Neural Machine Learning that translates the text into our familiar language, and it is called automatic translation.

3.2 Data Analytics using Python:

Python has become a staple in data science, allowing data analysts and other professionals to use the language to conduct complex statistical calculations, create data visualizations, build machine learning algorithms, manipulate and analyse data. and complete other data related tasks.

Python can build a wide range of different data visualizations, like line and bar graphs, pie charts, histograms, and 3D plots: Python also has a number of libraries that enable coders to write programs for data analysis and machine learning more quickly and efficiently, like TensorFlow and Keras.

There are various library modules used in the python programming language for the purpose of data analysis which includes modules like NumPy, Pandas, Scikit learn, Matplotlib, seaborn, tensorflow, missingno etc.

These libraries ease the user with the data handling, processing and as well as testing the same

Key topics:

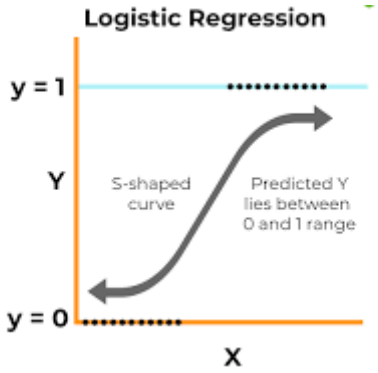
- Installing Python/Jupyter/Python on Windows Or Mac
- Python Basics (variables, strings, simple math, conditional logic, for loops, lists, tuples, dictionaries, etc.)
- Using the Pandas library to manipulate data (filtering and sorting data, combining files, GroupBy, etc.)
- Plotting data in Python using Matplotlib and Seaborn
- Multiple Linear Regression using Scikit-Learn
- Decision Trees using Scikit-Learn
- Random Forests (Scikit-Learn)

3.3 Machine learning Techniques

For fulfilling the objective of our Project as have used various techniques other than the government method for the prediction of the Fake Jobs Postings. We have tested the data frame with different methods and tested the accuracy using different Statistical tools and obtained the efficient Machine Learning technique for the prediction of Fake Jobs Postings with the variables we have tested. The Implemented methods are explained as follows

3.3.1 Logistic Regression

Logistic regression is a statistical and machine learning technique used for predicting the probability of a binary outcome, such as yes/no, pass/fail, or spam/not spam. Unlike linear regression, which predicts continuous values, logistic regression uses a logistic (sigmoid) function to map input features to a value between 0 and 1, representing the probability of belonging to a particular class. It is widely used in classification tasks because it provides interpretable results through coefficients that indicate the influence of each predictor variable. Logistic regression is simple yet powerful, making it a popular choice for applications in fields like healthcare, finance, marketing, and social sciences.



Algorithm:
Here's the algorithm for Logistic Regression in simple steps:

1. **Initialize Parameters**
   - Assign initial values (often zero or small random numbers) to the weights www and bias b.
2. **Compute Linear Combination**
   - For each data point, calculate z= w·x+b where x is the input features.
3. **Apply Sigmoid Function**
   - Convert z into a probability using the sigmoid function

$$\hat{y} = \frac{1}{1 + e^{-z}}$$

4. **Calculate the Loss (Cost Function)**
   - Use **binary cross-entropy loss**:

$$J(w, b) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$

5. **Update Parameters (Gradient Descent)**
   - Compute the gradients of the cost with respect to www and b.

- Update the parameters:

$$w = w - \alpha \frac{\partial J}{\partial w}, \quad b = b - \alpha \frac{\partial J}{\partial b}$$

Where, α is the learning rate.

6. **Repeat**
   - Continue steps 2–5 until the cost function converges (changes very little) or a maximum number of iterations is reached.
7. **Prediction**
   - For a new input x, calculate y^ using the sigmoid function.
   - Classify as **1 if y^≥0.5**, otherwise **0**.

Advantages:

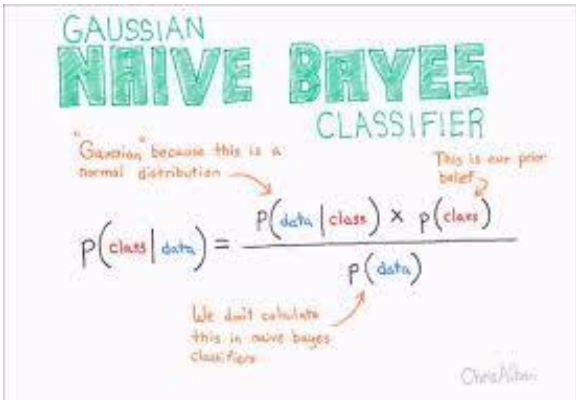
1. Simple & Easy to Interpret
2. Computationally Efficient
3. Probabilistic Output
4. Less Prone to Overfitting (with Regularization)
5. Works Well for Linearly Separable Data

Disadvantages:

1. Limited to Linear Boundaries.
2. Sensitive to Outliers.
3. Works poorly with raw, unscaled, or irrelevant features.
4. Not Suitable for Large Number of Features.
5. Assumes predictors are independent.

3.3.2 Naive Bayes

Naïve Bayes is a probabilistic classification algorithm based on Bayes' theorem. It assumes that all features are independent (hence the term *naïve*). Despite this simple assumption, it performs surprisingly well in many real-world problems. It calculates the probability of a data point belonging to each class and assigns it to the class with the highest probability. Naïve



Bayes is widely used for text classification (like spam filtering, sentiment analysis) and categorical data problems due to its simplicity and efficiency.

Algorithm:
Here's the **algorithm for Naïve Bayes** in simple steps:

1. **Prepare the Dataset**
   - Collect data and separate it into features X and labels Y.
   - Identify the number of classes C (e.g., spam/not spam).

2. **Calculate Prior Probabilities**
   - For each class Ck , compute the prior probability:

$$P(C_k) = \frac{\text{Number of samples in class } C_k}{\text{Total number of samples}}$$

3. **Calculate Likelihood (Conditional Probability)**
   - For each feature value xi given class Ck, compute:

     P(xi|Ck)
   - This depends on the type of data:

   a. **Gaussian NB:** assumes features follow a normal distribution.
   b. **Multinomial NB:** used for text data (word counts).
   c. **Bernoulli NB:** used for binary data (yes/no, present/absent).

4. **Apply Bayes' Theorem**
   - For a new sample X=(x1,x2,...,xn), calculate:

$$P(C_k|X) \propto P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$$

   - Multiply prior by all feature likelihoods (assuming independence).

5. **Choose the Class with the Highest Posterior Probability**
   - Assign the sample to the class Ck with the largest P(Ck|X).

Advantages:

1. Fast & Efficient.
2. Performs Well with Small Data.
3. Great for Text Classification.
4. Handles Irrelevant Features.

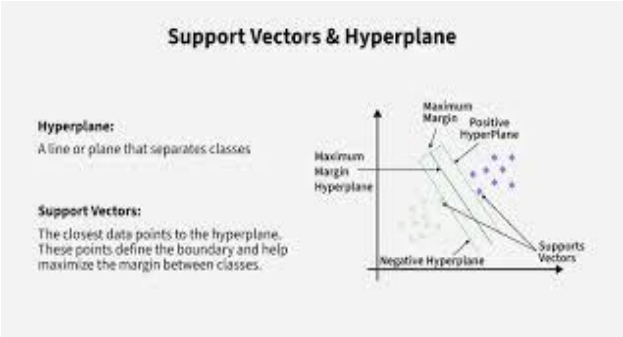
5. Simple & Easy to Implement.

Disadvantages:

1. Strong Independence Assumption.
2. Poor with Correlated Features.
3. Zero Probability Issue.
4. Not Suitable for Continuous Features Without Modification.
5. Less Flexible.

3.3.3 Support Vector Machine (SVM):

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification (and sometimes regression). It works by finding the best hyperplane that separates data points of different classes with the maximum margin (distance between the hyperplane and the nearest data points, called support vectors). SVM can handle linear and non-linear data using a technique called the kernel trick, which transforms data into higher dimensions where separation is possible.



Algorithm:

1. **Input Data**
   - Collect labelled training data (xi,yi) where yi ∈{−1,+1}.
2. **Find the Best Hyperplane**
   - Compute a hyperplane w·x+b=0 that separates the two classes.
   - The goal is to maximize the margin (distance between the hyperplane and the nearest data points).
3. **Identify Support Vectors**
   - Find the closest points to the hyperplane (support vectors). These determine the position and orientation of the hyperplane.
4. **Optimize (Convex Optimization)**
   - Solve for w and b by minimizing:

$$\min \frac{1}{2}\|w\|^2$$

subject to:

$$y_i (w \cdot x_i + b) \geq 1 \quad \forall i$$

- For non-linearly separable data, introduce slack variables (soft margin) and a penalty parameter (C).

5. **Use Kernel Trick (if needed)**
    - For complex data, apply a kernel function (e.g., polynomial, RBF) to map data into a higher dimension where it becomes separable.
6. **Prediction**
    - For a new point x, compute:

$$y = \text{sign}(w \cdot x + b)$$

- Classify as +1 or -1 based on which side of the hyperplane it lies.

Advantages:

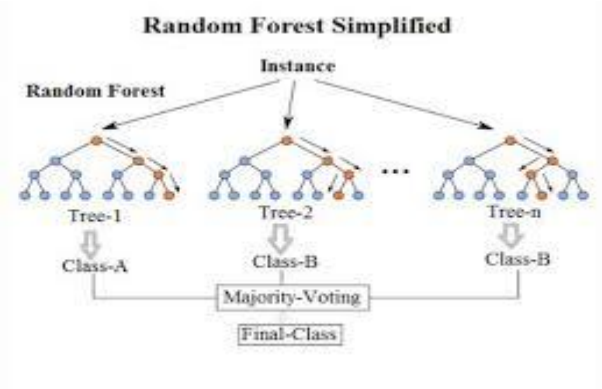
1. Effective for High-Dimensional Data.
2. Robust to Overfitting.
3. Works for Non-linear Data.
4. Good Generalization.
5. Memory Efficient.

Disadvantages:

1. Computationally Expensive.
2. Hard to Choose the Right Kernel.
3. Less Interpretable.
4. Not Ideal for Noisy Data.
5. No Probabilistic Output by Default.

3.3.4 Random forest:

Random Forest is a popular machine learning algorithm used for regression tasks. It works by constructing multiple decision trees during training and outputs the mean prediction of the individual trees for regression problems. It's effective for handling nonlinear relationships and is robust to overfitting.



Algorithm:
Here's the algorithm for Random Forest in simple steps:

1. **Input Data**
   - Start with a labelled dataset (X,Y).
2. **Bootstrap Sampling**
   - Randomly select samples with replacement from the dataset to create multiple subsets (one for each tree).
3. **Build Decision Trees**
   - For each subset:
     - Grow a decision tree using a random subset of features at each split (feature bagging).
     - No pruning is done — trees are grown fully.
4. **Repeat**
   - Repeat steps 2–3 for N trees (forest size).
5. **Aggregate Results**
   - **For classification:** Each tree votes for a class → final prediction is the majority vote.
   - **For regression:** Take the average of all tree outputs.
6. **Prediction for New Data**
   - Pass the new input through all trees.
   - Combine their predictions (vote/average) to get the final output.

Advantages:

1. High Accuracy.
2. Works Well with Large & High-Dimensional Data.
3. Robust to Noise & Outliers.
4. Handles Missing Data.
5. Feature Importance.

Disadvantages:

1. Computationally Expensive.
2. Less Interpretable.
3. May Overfit with Too Many Trees.
4. Memory-Intensive.
5. Not Great for Very Sparse Data.

3.4 Evaluation Metrics

Evaluation metrics were used to measure how well the models classified job postings as real or fake. The two main metrics applied were the confusion matrix and the classification report.

1.Confusion Matrix:

A confusion matrix is a performance evaluation tool used in classification problems to show how well a model's predictions match the actual labels.

It is especially useful when dealing with imbalanced datasets (like real vs fake job postings), where accuracy alone can be misleading.

It is structured as a **table** that compares predicted values with actual values and breaks them down into four categories:

- **True Positives (TP):** Correctly predicted positive cases (e.g., fake jobs correctly classified as fake).

- **True Negatives (TN):** Correctly predicted negative cases (e.g., real jobs correctly classified as real).

- **False Positives (FP):** Negative cases incorrectly predicted as positive (e.g., real jobs wrongly predicted as fake).

- **False Negatives (FN):** Positive cases incorrectly predicted as negative (e.g., fake jobs wrongly predicted as real).

this matrix helps identify **where the model is making errors** (e.g., over-predicting fake jobs or missing them) and provides the basis for calculating

2. Classification Metric:

For classification tasks, models predict a discrete class label, and evaluation focuses on how well the model can differentiate between these classes. Key metrics include:

Accuracy: The proportion of correct predictions (both positive and negative) over all predictions.

Accuracy= (True Positives + True Negatives) / (True Positives + True Negatives + False Positives + False Negatives)

Interpretation: Accuracy is intuitive but can be misleading when dealing with imbalanced datasets (where one class significantly outnumbers the other). For example, in a dataset with 95% of class A and 5% of class B, a model predicting only class A would have 95% accuracy but would fail to identify class B.

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

Precision= True Positives / (True Positives + False Positives)

Interpretation: Precision is important when the cost of a false positive is high (e.g., email spam detection, where a false positive means a valid email is marked as spam). High precision means fewer false positives.

Recall: The proportion of actual positive instances that are correctly identified.

Recall= True Positives / (True Positives + False Negatives)

Interpretation: Recall is important when the cost of a false negative is high (e.g., in medical diagnostics where missing a positive case could have severe consequences). High recall means fewer false negatives.

F1-Score: The harmonic means of precision and recall, which balances both metrics.

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

Interpretation: F1-Score is useful when you need a balance between precision and recall and when the class distribution is imbalanced. It's particularly useful for binary classification problems.

Support: The number of actual samples in each class (Real vs Fake) in the test dataset.

3.Macro Avg:

Macro average is an overall performance metric used in classification reports that calculates the simple (unweight) average of a chosen score, such as precision, recall, or F1-score, across all classes. In other words, it computes the metric for each class independently and then takes the average, giving equal importance to every class regardless of how many samples each has. This makes macro averaging especially useful when you want to evaluate how well a model performs across all classes equally, without being biased by class imbalance. In this project, macro averages were used to check if the models performed consistently on both real and fake job postings, even though the dataset was heavily skewed toward real jobs.

4. Weighted Avg:

Weighted average is an overall performance metric in a classification report that calculates the average of precision, recall, or F1-score across all classes, but unlike macro average, it weights each class's score by its support (the number of actual samples in that class). This means that classes with more samples contribute more to the final score, making it a more realistic measure of model performance on imbalanced datasets. In this project, since there were many more real job postings than fake ones, the weighted average provided a better sense of the model's true overall effectiveness, as it reflected how well the model performed considering the class distribution.

CHAPTER 4
SOFTWARE USED FOR DATA ANALYSIS

CHAPTER 4
SOFTWARE USED FOR DATA ANALYSIS

4.1 Introduction

4.1.1 Python Programming

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid at the Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program at the end maintenance. Python supports modules and packages, which encourages in program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

4.1.2 Importance of Python

Python's importance stems from its versatility, simplicity, and extensive ecosystem. As one of the most popular programming languages, Python facilitates rapid development across various domains, from web development and data science to artificial intelligence and automation. Its clear syntax and readability make it accessible to beginners while offering advanced features for experienced developers. With a vast array of libraries and frameworks, Python enables efficient problem-solving and fosters innovation. Its open-source nature fosters a collaborative community, driving continual improvement and adaptation. Overall, Python's significance lies in its power to streamline development processes, enhance productivity, and fuel technological advancements across industries.

4.1.3 Advantages of Python

Readability and simplicity: Python emphasizes readability and simplicity, which makes it easy to learn and understand, even for beginners. Its clean and straightforward syntax reduces the cost of program maintenance and development.

Large standard library: Python comes with a vast standard library that provides support for various tasks and functionalities, such as string operations, file I/O, networking, database interfaces, and much more. This extensive library helps developers accomplish complex tasks without having to write additional code from scratch.

Community and support: Python has a large and active community of developers, enthusiasts, and contributors who constantly work to improve the language. This vibrant community provides extensive support through forums, mailing lists, and online resources, making it easier for developers to find solutions to their problems and learn from others.

Portability: Python is a platform-independent language, meaning that Python code can run on various platforms and operating systems without any modifications. This portability makes it an ideal choice for developing crossplatform applications.

Flexibility and versatility: Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Its versatility allows developers to choose the most appropriate approach for their projects and switch between paradigms as needed.
Integration capabilities: Python seamlessly integrates with other programming languages and technologies, such as C/C++, Java, .NET, and more. This interoperability allows developers to leverage existing codebases and libraries written in other languages, enhancing productivity and efficiency.

4.1.4 Disadvantages of Python

Performance: Python is an interpreted language, which means it tends to be slower than compiled languages like C or C++. This can be a disadvantage in situations where high performance is critical, such as in real-time systems or high-frequency trading applications. While there are ways to optimize Python code (e.g., using libraries like NumPy or Cython), it may still not match the performance of compiled languages.
GIL (Global Interpreter Lock): Python has a Global Interpreter Lock, which means that only one thread can execute Python bytecode at a time. This can limit the effectiveness of multi-threading for CPU-bound tasks since threads cannot fully utilize multiple CPU cores simultaneously.
Mobile Development: While Python can be used for mobile development using frameworks like Kivy or BeeWare, it's not as commonly used as languages like Java or Swift for building native mobile applications. This can be a disadvantage if you're primarily focused on mobile development.
Memory Consumption: Python's dynamic typing and high-level abstractions can lead to higher memory consumption compared to languages like C or C++. This can be a concern in memory-constrained environments, such as embedded systems or when dealing with large scale data processing tasks.
Packaging and Distribution: Python's packaging and distribution ecosystem can be complex and fragmented. While tools like pip and virtual environments help manage dependencies, dealing with different versions of packages, compatibility issues, and ensuring consistent environments across different systems can be challenging, especially for beginners.

4.2 DATA ANALYSIS USING PYTHON

4.2.1 Basic Statistic

Frequency Counts:

In this project, frequency counts were used to calculate how often different categories, such as job titles, industries, locations, and employment types, appeared in the dataset. This provided a clear understanding of the distribution of these categorical variables and helped identify which categories were most common in real versus fake job postings. Such analysis is useful for spotting imbalances or unusual patterns, like specific industries or job roles that are overrepresented in fake postings.

Text Length Analysis:

Text length analysis involved measuring the number of words or characters in job descriptions and requirements. This was done to compare the distribution of text lengths between real and fake job postings. Fake postings often tend to be unusually short or excessively long, which can be a useful indicator for classification. By analyzing and visualizing these lengths, the project was able to detect these differences, making it an important part of exploratory analysis.

Word Frequency Analysis:

Word frequency analysis focused on identifying the most commonly used words in real versus fake job postings. This process involved tokenizing the text (splitting it into individual words) and counting their occurrences. By comparing the top words in each category, the project uncovered language patterns and keywords that could indicate fraudulent behavior, such as frequent use of terms like "immediate hire," "urgent," or "free registration" in fake postings.

Sentiment Analysis:

Sentiment analysis was performed using the TextBlob library to calculate the polarity of job descriptions, which indicates whether the tone of the text is positive, negative, or neutral. This helped explore how the sentiment of fake job postings differed from real ones. For instance, fake postings may use overly enthusiastic or persuasive language to lure applicants, while real postings tend to maintain a more neutral or professional tone.

4.2.2 Data Visualisations

Bar Chart

A bar chart is a graphical representation of data that uses rectangular bars of varying lengths to represent the frequency, count, or proportion of different categories or values in a dataset. Each bar typically corresponds to a specific category, with the length of the bar proportional to the magnitude of the data it represents. Bar charts are commonly used to compare and visualize discrete categories or groups of data, making it easy to identify patterns, trends, and relationships within the dataset. They are effective for presenting categorical data and are widely used in presentations, reports, and data visualization applications.

Histogram

A histogram is a graphical representation of the distribution of numerical data. It consists of a series of bars, where each bar represents the frequency or relative frequency of data within a certain range or interval. The horizontal axis typically represents the range of values, divided into bins or intervals, while the vertical axis represents the frequency or relative frequency of occurrences within each bin. Histograms provide a visual depiction of the shape, centre, and spread of a dataset, making it easy to identify patterns, outliers, and underlying distributions in the data. They are commonly used in exploratory data analysis and statistical inference.

Count plot

A count plot is a simple yet powerful visualization used to display the frequency of categories in a dataset. It creates a bar chart where the height of each bar represents the number of occurrences of a particular category, making it easy to compare different groups at a glance. Count plots are especially useful for exploring categorical variables, such as job titles, employment types, or locations, as they quickly reveal which categories are most or least common. In your project, count plots were used to visualize the distribution of employment types, industries, and locations for real and fake job postings, helping to identify patterns and imbalances in the data. This makes them an essential part of exploratory data analysis (EDA), as they provide an immediate understanding of categorical data distribution.

4.2.3 Packages used

Numpy

NumPy is a powerful Python library for numerical computing. It provides highperformance multidimensional array objects and tools for working with these arrays efficiently. Numpy's arrays facilitate mathematical operations on large datasets, making it popular in fields like data science, machine learning, and scientific computing. Additionally, NumPy offers a wide range of mathematical functions, linear algebra operations, and random number generation capabilities. Its efficient array operations and broadcasting capabilities make it a fundamental building block for many Python-based numerical computing tasks.

Pandas

Pandas is a powerful open-source Python library for data manipulation and analysis. It provides easy-to-use data structures like DataFrame and Series, which allow for efficient handling of structured data. With Pandas, users can perform operations such as filtering, grouping, sorting, and aggregation on datasets, making it a popular choice for data wrangling tasks. It also integrates well with other libraries in the Python ecosystem, such as NumPy and Matplotlib, enabling seamless data analysis and visualization workflows.

Matplotlib

Matplotlib is a popular Python library used for creating static, interactive, and animated visualizations. It offers a wide range of plotting functionalities for generating plots, charts, histograms, and more. Matplotlib provides a high degree of customization and flexibility, allowing users to create publicationquality graphics with ease. It integrates seamlessly with other libraries such as NumPy and Pandas, making it a powerful tool for data visualization and analysis in fields like data science, machine learning, finance, and scientific research.

Scikit-learn

Scikit-learn is a popular machine learning library in Python, providing a simple and efficient toolset for data mining and analysis. It offers various supervised and unsupervised learning algorithms, including classification, regression, clustering, dimensionality reduction, and model selection. Scikitlearn is built on NumPy, SciPy, and matplotlib, making it easy to integrate into existing Python data analysis workflows. Its user-friendly interface and extensive documentation make it a preferred choice for both beginners and experienced practitioners in the field of machine learning and data science.

Seaborn

Seaborn is a powerful Python visualization library built on top of Matplotlib, designed for creating aesthetically appealing and informative statistical graphics. It provides high-level functions for easily visualizing complex datasets, such as scatter plots, box plots, violin plots, and pair plots, making it particularly useful for exploratory data analysis. Seaborn integrates well with Pandas DataFrames and includes built-in themes and color palettes to enhance readability. It also supports advanced statistical visualizations like regression plots and heatmaps for correlation analysis. By simplifying the process of data visualization, Seaborn helps analysts and data scientists gain deeper insights into their data with minimal code.

Re and String

The re (regular expressions) and string modules were used extensively for text preprocessing. The re module helped in identifying and removing unwanted patterns such as special characters, digits, URLs, and extra spaces from job descriptions and requirements, making the text cleaner and more consistent for analysis. On the other hand, the string module provided convenient constants like `string.punctuation` and `string.ascii_letters`, which simplified the process of filtering out punctuation and non-alphabetic characters. Together, these packages played a crucial role in preparing the textual data for further processing, ensuring that only meaningful and relevant content was retained for feature extraction and model training.

NLTK

the Natural Language Toolkit (NLTK) library was used for text preprocessing and natural language processing (NLP) tasks. It provided essential tools for cleaning and transforming textual data, such as removing stopwords (common words like “and”, “the”, “is” that do not add meaning), tokenizing text (splitting sentences into individual words), and lemmatization (reducing words to their base or root form, e.g., “running” to “run”). These steps helped standardize the job descriptions and requirements, reducing noise and improving the quality of textual features. By using NLTK, the project ensured that the text data was normalized and ready for feature extraction techniques like TF-IDF, ultimately enhancing the performance of machine learning models for detecting fake job postings.

Joblib

The joblib library was used for **model persistence**, which means saving and loading trained machine learning models efficiently. After training classifiers like Logistic Regression, Naive Bayes, SVM, and Random Forest on the processed job posting data, joblib allowed these models to be stored as files without needing to retrain them each time. This not only saves computational time but also makes it easier to deploy the models for real-world use, such as detecting fake job postings on new data. By using joblib, the project ensured a smooth workflow for saving trained models and reusing them whenever needed.

4.2.4 Machine Learning Methods

Logistic Regression

Syntax:

```
from sklearn.linear_model import LogisticRegression

# Create the model
model = LogisticRegression()

# Train the model
model.fit(X_train, y_train)

# Make predictions
predictions = model.predict(X_test)
```

Naïve Bayes:

Syntax:

```
from sklearn.naive_bayes import MultinomialNB

# Create the model
model = MultinomialNB()

# Train the model
model.fit(X_train, y_train)

# Make predictions
predictions = model.predict(X_test)
```

Support Vector Machine (SVM)

Syntax:

```
from sklearn.svm import LinearSVC

# Create the model
model = LinearSVC()

# Train the model
model.fit(X_train, y_train)

# Make predictions
predictions = model.predict(X_test)
```

Random Forest

Syntax:

```
from sklearn.ensemble import RandomForestClassifier

# Create the model
model = RandomForestClassifier()

# Train the model
model.fit(X_train, y_train)

# Make predictions
predictions = model.predict(X_test)
```

CHAPTER 5
IMPLEMENTATION USING PYTHON

CHAPTER 4
IMPLEMENTATION USING PYTHON

5.1 Importing packages & Dataset

```
import pandas as pd
import numpy as np
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import joblib

# Download necessary NLTK data
nltk.download('stopwords')
nltk.download('wordnet')

# Load dataset
df = pd.read_csv(C:/Users/Dell/Downloads/fake_job_postings (1).csv")
```

5.2 Basic Statistics

```
# Basic statistical summary
print(df.describe(include='all'))

# Checking class distribution (Real vs Fake)
print(df['fraudulent'].value_counts())

# Adding new feature: length of requirements text
df['requirements'] = df['requirements'].fillna('')
df['requirements_len'] = df['requirements'].apply(len)

# Plot distribution of requirements length
plt.figure(figsize=(8, 6))
```

```
df[df.fraudulent == 0]['requirements_len'].plot(
   bins=35, kind='hist', color='blue', alpha=0.6, label='Real Positions')
df[df.fraudulent == 1]['requirements_len'].plot(
   kind='hist', color='red', alpha=0.6, label='Fake Positions')
plt.title('Requirements Length Distribution')
plt.legend()
plt.show()
```

5.3 Text Pre-Processsing

```
# Combine relevant text columns
df['text'] = df[['title', 'company_profile', 'description', 'requirements',
'benefits']].fillna('').agg(' '.join, axis=1)

# Text cleaning function
import re, string
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

def clean_text(text):
   text = text.lower()  # Lowercase
   text = re.sub(r"http\S+|www\S+|https\S+", '', text)  # Remove URLs
   text = re.sub(r'\@\w+|\#', '', text)  # Remove mentions & hashtags
   text = re.sub(r"[^a-zA-Z]", " ", text)  # Keep only alphabets
   text = text.translate(str.maketrans('', '', string.punctuation))  # Remove punctuation
   stop_words = set(stopwords.words('english'))  # Stopwords
   lemmatizer = WordNetLemmatizer()
   tokens = text.split()
   return ' '.join([lemmatizer.lemmatize(w) for w in tokens if w not in stop_words])

# Apply cleaning
df['clean_text'] = df['text'].apply(clean_text)
```

5.4 Data Visulization

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid")
fig, axs = plt.subplots(2, 2, figsize=(16, 12))

# Count Plot :1. Distribution of Real vs Fake Job Postings
sns.countplot(data=df, x='fraudulent', ax=axs[0, 0], palette='Set2')
axs[0, 0].set_title('Distribution of Real vs Fake Job Postings')
axs[0, 0].set_xticklabels(['Real (0)', 'Fake (1)'])
axs[0, 0].set_xlabel("Fraudulent")
```

```
axs[0, 0].set_ylabel("Count")

# 2. Top 10 Industries with Most Fake Job Postings
if 'industry' in df.columns:
    top_fake_industries = df[df['fraudulent'] == 1]['industry'].value_counts().head(10)
    top_fake_industries.plot(kind='bar', ax=axs[0, 1], color='orange')
    axs[0, 1].set_title('Top 10 Industries with Most Fake Job Postings')

# 3. Employment Type vs Fraudulence
if 'employment_type' in df.columns:
    sns.countplot(data=df, x='employment_type', hue='fraudulent', ax=axs[1, 0],
palette='Set1')
    axs[1, 0].set_title('Employment Type vs Fraudulence')

# 4. Telecommuting vs Fraudulence
if 'telecommuting' in df.columns:
    sns.countplot(data=df, x='telecommuting', hue='fraudulent', ax=axs[1, 1],
palette='coolwarm')
    axs[1, 1].set_title('Telecommuting vs Fraudulence')

fig.tight_layout()
plt.show()

# Extended: Top Job Titles for Real vs Fake Jobs
data_real = df[df.fraudulent == 0]
data_fake = df[df.fraudulent == 1]
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 6))
data_real['title'].value_counts().head(20).plot(kind='barh', ax=ax1, color='green')
data_fake['title'].value_counts().head(20).plot(kind='barh', ax=ax2, color='red')
ax1.set_title('Top Job Titles - Real')
ax2.set_title('Top Job Titles - Fake')
plt.tight_layout()
plt.show()

# 5. Histogram: Length of Requirements Text
df['requirements'] = df['requirements'].fillna('')
df['requirements_len'] = df['requirements'].apply(len)
plt.figure(figsize=(8, 6))
df[df.fraudulent == 0]['requirements_len'].plot(bins=35, kind='hist', color='blue',
label='Real', alpha=0.6)
df[df.fraudulent == 1]['requirements_len'].plot(kind='hist', color='red', label='Fake',
alpha=0.6)
plt.title('Requirements Length Distribution')
plt.legend()
plt.show()
```

5.5 Feature Extraction:

```
# Step 4: Vectorization (TF-IDF)
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1, 2))
X = vectorizer.fit_transform(df['clean_text'])
y = df['fraudulent']
```

5.6 Machine Learning and NLP code

```
# Step 5: Train-Test Split
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 6: Model Training
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

models = {
   "Logistic Regression": LogisticRegression(max_iter=1000),
   "Naive Bayes": MultinomialNB(),
   "SVM": LinearSVC(),
   "Random Forest": RandomForestClassifier(n_estimators=100)
}

for name, model in models.items():
   print(f"\nTraining {name}.")
   model.fit(X_train, y_train)
   y_pred = model.predict(X_val)
   print(confusion_matrix(y_val, y_pred))
   print(classification_report(y_val, y_pred))
```

5.7 Keyword Extraction

```
import numpy as np
def top_keywords(class_label, n=20):
   class_indices = df[df['fraudulent'] == class_label].index
   tfidf_class = vectorizer.transform(df.loc[class_indices, 'clean_text'])
   mean_tfidf = np.mean(tfidf_class.toarray(), axis=0)
   top_n_ids = np.argsort(mean_tfidf)[-n:]
```

```
    words = np.array(vectorizer.get_feature_names_out())[top_n_ids]
    print(f"\nTop keywords for {'Fake' if class_label==1 else 'Real'} Jobs:")
    print(list(words[::-1]))

top_keywords(0)
top_keywords(1)
```

5.8 Prediction code

```
# Step 9: Demo Testing Data
demo_jobs = [
    "Earn $5000/week working from home! No experience required!",
    "We are hiring a software engineer with 3+ years of experience in Python and Django.",
    "Congratulations! You've been shortlisted for a high-paying job. Click here to claim.",
    "Join our marketing team. Looking for energetic, creative individuals with a passion for branding.",
    "Part-time typist needed urgently. Work 2 hours a day and earn $3000/week. Apply now!",
    "A reputed MNC is hiring Data Analysts. Must be proficient in SQL and Excel."
    # ... (more test samples)
]

# Clean and vectorize demo jobs
demo_cleaned = [clean_text(text) for text in demo_jobs]
demo_vectorized = vectorizer.transform(demo_cleaned)

# Predict using the best model (choose the best performing one)
best_model = models["SVM"]  # or "Logistic Regression", "Naive Bayes"
preds = best_model.predict(demo_vectorized)

# Display results
for i, text in enumerate(demo_jobs):
    print(f"\nJob: {text}")
    print(f"Prediction: {'Fake' if preds[i] == 1 else 'Real'}")
```

5.9 Evaluation Metric code

```
from sklearn.metrics import classification_report, confusion_matrix

# Train & Evaluate Models
for name, model in models.items():
    print(f"\nTraining {name}.")
```

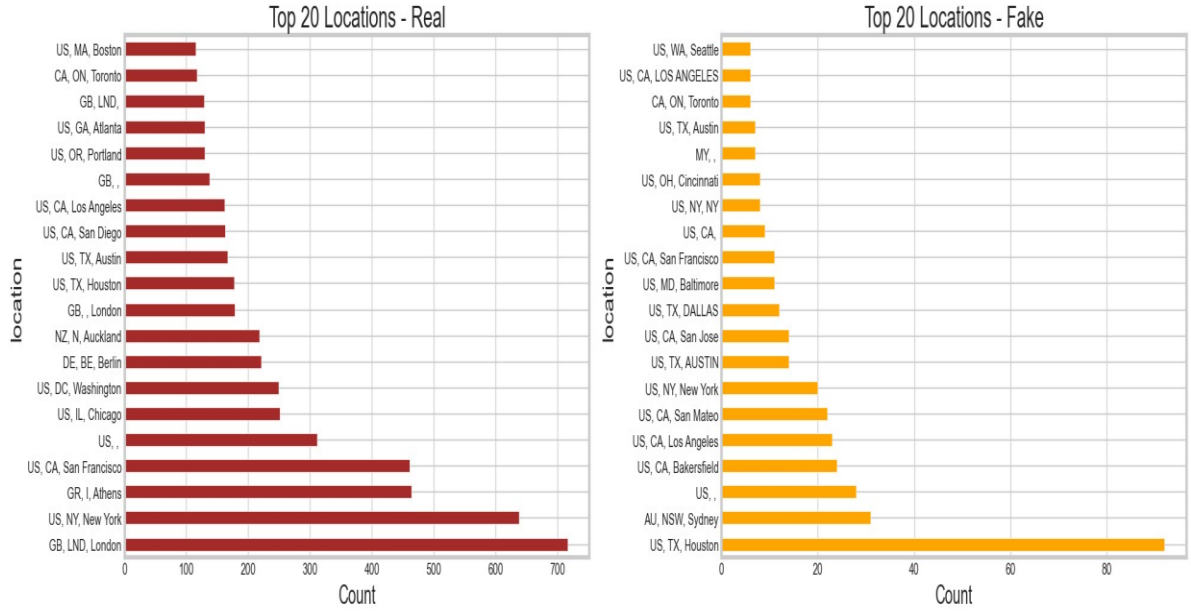
```
model.fit(X_train, y_train)
y_pred = model.predict(X_val)

# Confusion Matrix
print(confusion_matrix(y_val, y_pred))

# Classification Report
print(classification_report(y_val, y_pred)
```

CHAPTER 6
DATA ANALYSIS
&
INTERPRETATION

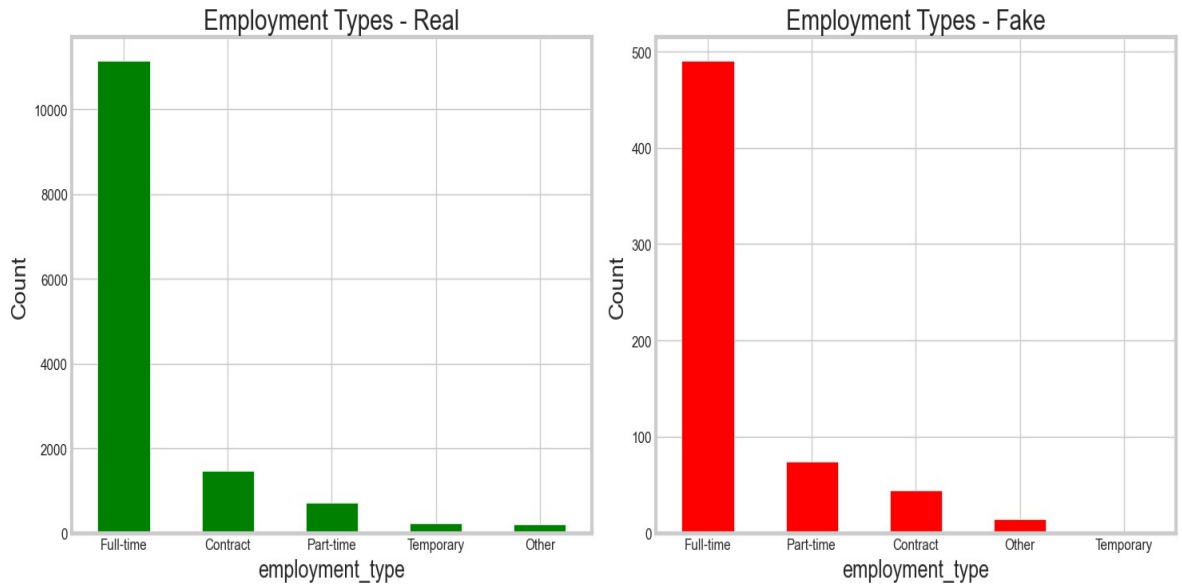
CHAPTER 6
DATA ANALYSIS & INTERPRETATION

6.1 Data Visualization

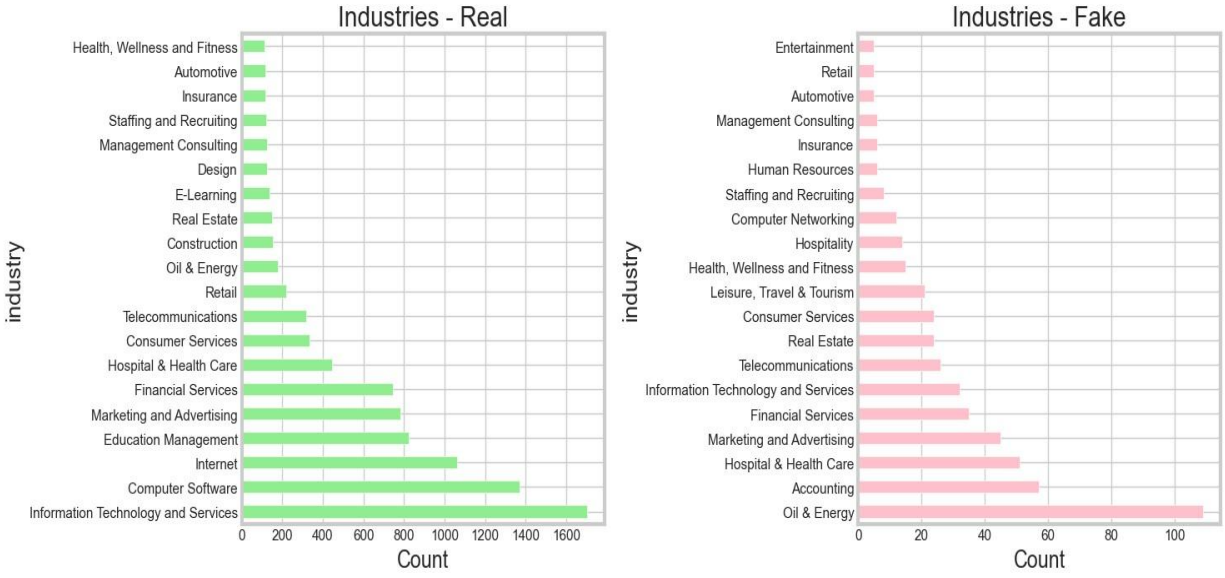
Interpretation: The bar chart shows the distribution of real vs. fake job postings. Most job postings are real (class 0), with around 17,000 entries, while fake job postings (class 1) are significantly fewer, roughly around 1,000. This indicates a highly imbalanced dataset where genuine postings dominate. Such imbalance may affect model performance and requires special handling during analysis or model training.



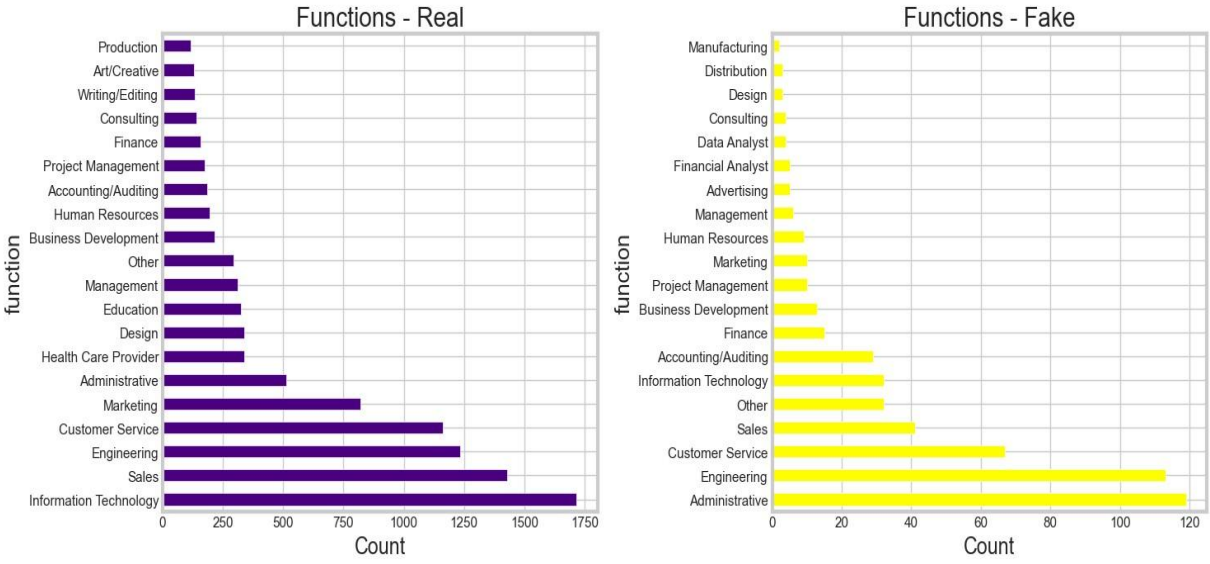
Interpretation: The charts display the top 20 locations for real and fake job postings. For real postings, London, New York, Athens, and San Francisco are the most common locations. In contrast, fake postings are concentrated in fewer places, with Houston, Sydney, and Los Angeles appearing most frequently. This suggests that fake postings are more localized compared to the wider distribution of real postings.



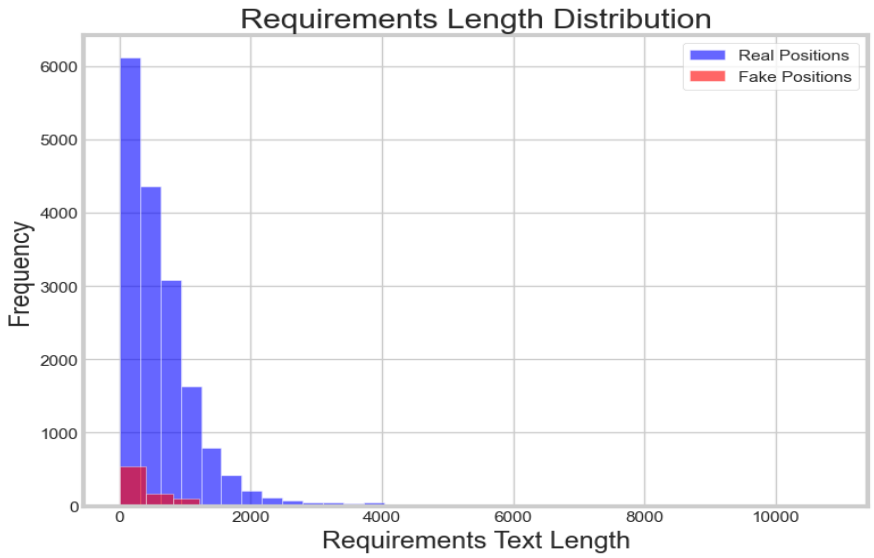
Interpretation: The charts show the distribution of employment types for real and fake job postings. Full-time roles dominate both categories, with over 11,000 in real postings and nearly 500 in fake ones. Contract and part-time positions appear more frequently in real postings than in fake ones. This indicates that fake job postings mostly target full-time roles, likely to attract more applicants.



Interpretation: The bar charts compare the distribution of industries in real vs. fake job postings. In real postings, "Information Technology and Services" and "Computer Software" dominate, indicating genuine demand. In fake postings, "Oil & Energy" and "Accounting" are most frequent, suggesting these industries are more targeted by fraudulent listings. This contrast highlights how fraudsters may exploit specific high-paying or technical sectors to appear legitimate.



Interpretation: The charts show job function distribution in real vs. fake job postings. In real postings, "Information Technology," "Sales," and "Engineering" are the most common roles. In fake postings, "Administrative," "Engineering," and "Customer Service" dominate, indicating fraudsters often mimic general or technical roles to deceive job seekers. The stark difference highlights targeted manipulation in certain job functions.



Interpretation: The histogram shows that real job postings (blue) tend to have longer and more detailed requirements sections, with a wide distribution peaking around 500–1000

characters. In contrast, fake job postings (red) are generally shorter, often under 500 characters. This suggests that fake listings may lack specificity or detail, making requirement length a potential indicator of authenticity.

6.2 Observation:

Real jobs keywords

Sl. No.	Keyword
1	experience
2	team
3	work
4	customer
5	service
6	company
7	business
8	sale
9	client
10	product
11	job
12	marketing
13	development
14	design
15	skill
16	amp
17	project
18	new
19	management
20	year

Fake jobs keywords

SI.No	Keyword
1	work
2	service
3	data entry
4	entry
5	position
6	customer
7	skill
8	amp
9	home
10	experience
11	data
12	time
13	project
14	job
15	per
16	system
17	earn
18	company
19	must
20	product

Interpretation: The keyword lists show clear differences between real and fake job postings. Real jobs emphasize professional terms like **"experience," "team," "business," "development,"** and **"management,"** indicating a focus on qualifications and role clarity. Fake jobs frequently use vague or generic terms such as **"data entry," "home," "earn," "per,"** and **"must,"** suggesting they often target less experienced applicants with enticing but ambiguous offers. Both lists share some common terms like **"work," "customer," "project," "job,"** and **"company,"** but context and additional keywords help differentiate authenticity.

6.3 Machine Learning:

Logistic Regression:

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	3395	0
Actual 1	99	82

Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.97	1	0.99	3395
1	1	0.45	0.62	181

Metric Score

Accuracy	0.97
Macro Avg F1	0.8
Weighted Avg F1	0.97

Naïve Bayes:

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	3393	2
Actual 1	108	73

Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.97	1	0.98	3395
1	0.97	0.4	0.57	181

Metric Score

Accuracy	0.97
Macro Avg F1	0.78
Weighted Avg F1	0.96

Support Vector Machine (SVM):

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	3391	4
Actual 1	52	129

Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.98	1	0.99	3395
1	0.97	0.71	0.82	181

Metric Score

Accuracy	0.98
Macro Avg F1	0.91
Weighted Avg F1	0.98

Random Forest:

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	3395	0
Actual 1	70	111

Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.98	1	0.99	3395
1	1	0.61	0.76	181

Metric Score

Accuracy	0.98
Macro Avg F1	0.88
Weighted Avg F1	0.98

Best Model based on performance:

Model	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)
Logistic Regression	1	0.45	0.62
Naive Bayes	0.97	0.4	0.57
SVM	0.97	0.71	0.82
Random Forest	1	0.61	0.76

Interpretation: The table presents the performance of four models—Logistic Regression, Naive Bayes, SVM, and Random Forest—for detecting fake job postings, evaluated using Precision, Recall, and F1-Score for Class 1 (fake jobs).

Logistic Regression and Random Forest both have perfect precision (1.0), meaning they never wrongly classify real jobs as fake. However, Logistic Regression has low recall (0.45), missing over half of the fake jobs, resulting in a moderate F1-score of 0.62. Naive Bayes also shows high precision (0.97) but the lowest recall (0.40), leading to the weakest F1-score (0.57), making it the least effective.

SVM stands out with a high precision of 0.97 and the highest recall (0.71), successfully identifying most fake jobs while minimizing false positives. This balance gives it the best F1-score of 0.82, making it the most effective model overall. Random Forest also performs well with a decent recall (0.61) and an F1-score of 0.76, ranking second.

In summary, SVM is the best model due to its balanced and high performance, followed by Random Forest. Logistic Regression and Naive Bayes are less effective due to their lower recall.

6.4 Testing Data (Prediction):

Sl. No.	Job Description	Prediction
1	Earn $5000/week working from home! No experience required!	Fake
2	We are hiring a software engineer with 3+ years of experience in Python and Django.	Real
3	Congratulations! You've been shortlisted for a high-paying job. Click here to claim.	Real
4	Join our marketing team. Looking for energetic, creative individuals with a passion for branding.	Real
5	Part-time typist needed urgently. Work 2 hours a day and earn $3000/week. Apply now!	Fake
6	A reputed MNC is hiring Data Analysts. Must be proficient in SQL and Excel.	Real
7	Limited slots! Work-from-home job opportunity. Immediate joining with great perks.	Fake
8	We are hiring a project manager with PMP certification and 5+ years of relevant experience.	Real
9	Get rich quick by working online. No skills required. Join our network and start earning today.	Real
10	Looking for a dedicated HR professional to manage recruitment and onboarding in our New York office.	Real
11	Receive $1000 bonus after signing up! Just fill out a form and you're in!	Fake
12	We are looking for a UI/UX Designer with at least 2 years of hands-on experience using Figma and Adobe XD.	Real
13	Click this link to win a free iPhone and secure a remote job at the same time!	Real
14	Join our finance department as a certified accountant. Prior audit experience preferred.	Real
15	Urgent requirement! Make money fast by completing simple online tasks!	Real
16	Hiring content writers to develop blog articles on technology and health topics. Must have excellent grammar.	Real
17	No resume required! Get access to a secret job market with huge payouts!	Real
18	Vacancy: Full-stack Developer with React and Node.js experience. Competitive salary and benefits.	Real
19	We offer instant interviews. Just send your details and start earning from home today!	Real
20	Customer Service Representative needed for a BPO in Mumbai. Fluent English required.	Real
21	Work from the beach and make $8000/month. Limited positions available. Apply fast!	Real
22	Hiring interns for social media management. Certificate and stipend provided.	Real
23	Earn while you sleep. Our affiliate program guarantees passive income with no investment!	Real
24	Research Assistant position available at the University of California. Must have experience in data analysis.	Real
25	Get your dream job without an interview. We guarantee placement!	Real
26	Backend Developer needed with experience in Python, Flask, and PostgreSQL. Remote OK.	Real
27	This is not a scam! Make thousands per week by joining our WhatsApp group!	Real
28	Business Analyst required for Fortune 500 company. Strong Excel and Power BI skills necessary.	Real
29	Guaranteed job placement if you pay a small registration fee today!	Real
30	Graphic Designer wanted to work on e-commerce creatives. Portfolio required.	Real

Interpretation: The image displays a portion of the testing dataset used for evaluating a fake job posting detection model. Each row in the dataset represents an individual job listing and includes various features such as location, department, salary range, company profile, job description, requirements, benefits, and more. These attributes provide essential context and content for each job post. The dataset also includes binary indicators like telecommuting status and employment type, along with qualifications such as required experience and education level. Most importantly, the dataset contains a target column labelled "fraudulent," where a value of 0 indicates a real job and 1 signifies a fake one. This testing data plays a crucial role in assessing the model's ability to generalize and accurately detect fake job postings based on patterns observed during training. It reflects a realistic scenario where the model must evaluate new, unseen job entries and classify them correctly based on the given features.

CHAPTER 7
CONCLUSION
&
FUTURE SCOPE

CHAPTER 7
CONCLUSION & FUTURE SCOPE

7.1 Conclusion:

This project has successfully demonstrated the practical application and effectiveness of Natural Language Processing (NLP) and Machine Learning (ML) techniques in addressing the growing concern of fake job postings on online platforms. By leveraging a real-world dataset comprising 17,880 job postings sourced from GitHub, we developed a comprehensive pipeline that spans from data pre-processing to model evaluation and prediction.

The initial phase involved rigorous data cleaning and pre-processing, which included handling missing values, removing duplicates, and standardizing categorical data. Key feature engineering steps such as text cleaning, tokenization, stemming, and lemmatization were carried out to convert unstructured text into a structured format suitable for machine learning models.

Following pre-processing, we conducted an in-depth exploratory data analysis to uncover linguistic and structural patterns that distinguish fake job listings from real ones. This analysis revealed that fake postings often contain exaggerated claims, urgent call-to-actions (e.g., "Earn $5000/week", "No experience required"), and frequently lack credible information such as company descriptions or logos. In contrast, genuine postings emphasized teamwork, technical skills, professional experience, and detailed job roles.

Using keyword frequency analysis, we identified the most common terms associated with real and fake jobs. For instance:

Real jobs emphasized words like experience, team, customer, development, and management.

Fake jobs featured terms like data entry, home, earn, per, and system, indicating promotional or vague content.

To build a robust predictive framework, we trained and evaluated four machine learning models:

1. Logistic Regression
2. Naive Bayes
3. Support Vector Machine (SVM)
4. Random Forest

Each model was tested using a labelled dataset, and their performance was assessed using metrics such as precision, recall, F1-score, and accuracy. Among these, the Support Vector Machine (SVM) outperformed the others, achieving an accuracy of 98% and an F1-score of 0.82 for the fraudulent class. This indicates that SVM was particularly effective in distinguishing between fake and genuine job postings, even when faced with imbalanced class distributions.

To further validate the model's generalizability, we applied the trained SVM model to new, unseen job postings. The model successfully predicted the authenticity of these jobs, correctly flagging suspicious entries and confirming legitimate ones. These results demonstrate the model's ability to make accurate predictions in real-world scenarios.

Finally, the study affirms that automated classification systems, when powered by advanced NLP and ML algorithms, can play a crucial role in enhancing the safety and trustworthiness of online recruitment platforms. By integrating such models, job portals and HR systems can proactively detect and filter out fraudulent listings, thereby:

- Protecting job seekers from scams and data theft.
- Enhancing user confidence in job platforms.
- Reducing the administrative burden of manual screening.

In conclusion, this project highlights the transformative potential of AI-powered tools in securing digital job markets. With continuous improvements and real-time model deployment, such systems can significantly reduce the exposure of users to fraudulent job opportunities and foster a more reliable digital employment ecosystem.

7.2 Future Scope:

While this project has demonstrated promising results in detecting fake job postings using NLP and ML techniques, several areas offer opportunities for further enhancement and expansion:

- Real-Time Detection System

Future work can focus on deploying the model into a real-time job portal environment using web frameworks like Flask or FastAPI. This would enable immediate detection and flagging of fake postings as soon as they are submitted.

- Model Generalization Across Platforms

The current model is trained on a dataset from a single source. To improve generalizability, future studies can involve multi-platform datasets (e.g., LinkedIn, Indeed, Glassdoor) to capture broader patterns of fraudulent activity.

- Incorporating Deep Learning Techniques

Advanced deep learning models such as BERT, RoBERTa, or LSTM networks can be explored to capture deeper contextual and semantic relationships in job descriptions, potentially improving performance on subtle or cleverly disguised scams.

- User Feedback Loop

Introducing a feedback system where users can report suspicious job posts would help in updating the dataset continuously, retraining models with fresh data, and improving long-term accuracy.

- Multilingual Support

Expanding the system to support job listings in multiple languages can help detect fraudulent posts across global platforms and widen the scope of deployment.

- Explainable AI (XAI)

Integrating explain ability techniques can make model predictions more transparent. This helps users and administrators understand why a job post was flagged, which builds trust in the system.

- Integration with HR and Recruitment Systems

The model can be embedded into Applicant Tracking Systems (ATS) or HR software to pre-screen postings before they go live, thereby reducing manual screening efforts.

- Detection of Other Fraudulent Activities

The same methodology could be adapted to detect other types of online fraud, such as fake product reviews, phishing emails, or scam advertisements, using similar NLP pipelines.

CHAPTER 8
BIBLOGRAPHY

CHAPTER 8
BIBLOGRAPHY

Biblography:

Data: https://github.com
ICDIS 2023, published early 2024, “Real and Fake Job Classification Using NLP and Machine Learning Techniques”
Detection of Fake Job Postings by Utilizing Machine Learning and NLP Approaches”
Johnson and Patel, "Feature Engineering for Fake Job Posting Detection"
pythonspot.com
diveintopython3.net
python.swaroopch.com
learnpython.org
Michal Jaworski, Tarek Zindé (2021):" Expert Python Programming", 4th Edition, Packt Publishing Limited.
Manaranjan Pradhan, U Dinesh Kumar (2019):" Machine Learning Using Python.
Python: Data Analytics and Visualization-2017-Ashish kumar, Kirthi Raman, PACKT publications.
Wes McKinney, (2011), "Python for Data Analysis", O’Reilly (Third Edition).
Chen and Wu, "Ensemble Learning for Fake Job Posting Detection"
URL: https://tijer.org/tijer/papers/TIJER2309114.pdf
Smith et al. "Detecting Fake Job Postings: A Natural Language Processing Approach"

Reviews:

1. Topic Modeling Combined with Classification
    1. Author: Anurag Kumar
    2. Overview: Employed topic modeling to uncover latent themes in job postings, followed by classification models to predict fraudulent listings. This approach provided insights into the underlying topics associated with fake job postings.
    3. Link: GitHub
2. Deep Learning Classifier with STEM vs. Non-STEM Analysis
    1. Author: jylee1357
    2. Overview: Implemented a deep learning classifier to differentiate between fake and real job postings. Additionally, analyzed the prevalence of fraudulent postings in STEM versus non-STEM fields, revealing fake job posting patterns.
    3. Link: GitHub

3. Comprehensive EDA with Multiple Classifiers
    1. Author: Faizan Ali
    2. Overview: Conducted extensive exploratory data analysis to identify patterns in fraudulent postings, such as shorter descriptions and lack of company logos. Tested various classifiers, including Logistic Regression, SVM, and KNN
    3. Link: Medium
4. LSTM-Based Neural Network for Text Classification
    1. Author: Vishnu Karande
    2. Overview: Applied Long Short-Term Memory (LSTM) networks to capture sequential dependencies in job descriptions, enhancing the detection of fraudulent postings. Compared performance with traditional machine learning algorithms.
    3. Link: GitHub