AN INTERNSHIP PROJECT

ON

AI RESUME ANALYZER

Team Members :-

Rishi Shankar

Shambhavi

ACKNOWLEDGMENT

We would like to express our sincere gratitude to **Mr. Ganesh Nagu Doddi (Founder and CEO, Brain O Vision solutions India Pvt Ltd.)** for providing me an opportunity to do my internship for this semester. We would also like to thank my mentor **Mr. Nagoor Shaik (Technical Lead, Brain O Vision Solutions India Pvt Ltd.)** for his constant support throughout this project journey and also like to thank my team member for their support during the period of our project work.

ABSTRACT

The AI-Powered Resume Analyzer is an intelligent, interactive web application designed to automate and enhance the resume screening process. Developed using Python and Streamlit, this system leverages natural language processing (NLP) and machine learning techniques to parse and analyze resumes submitted in PDF format. Its core objective is to assist job seekers by providing instant, data-driven feedback on their resumes, as well as personalized recommendations for skills development, courses, and improvements, thereby improving their chances of securing a desired job role.

Upon uploading a resume, the application extracts key candidate details—such as name, contact information, and skills—using the pyresparser and pdfminer libraries. The resume is then evaluated across various parameters including completeness (presence of key sections like Objectives, Projects, Achievements, etc.), number of pages, and relevance of listed skills. A resume score is computed to give candidates a quantitative measure of resume strength.

A standout feature of the system is its ability to predict the user's potential job domain—such as Data Science, Web Development, Android Development, iOS Development, or UI/UX Design—based on extracted skills. It cross-references these skills against predefined keyword clusters and, depending on the match, provides personalized course and certification suggestions to fill knowledge gaps. The application also displays real-time skill tags and resume improvement tips directly

in the user interface.

The backend uses a MySQL database to log user details, analysis results, and recommendations, which can be reviewed later in the admin dashboard. The admin panel also provides graphical summaries through pie charts to visualize trends in predicted job domains and user experience levels among all analyzed resumes.

By automating resume screening and enhancing it with smart recommendations, this project not only empowers job seekers but also provides insightful analytics for recruiters and administrators. The AI-Powered Resume Analyzer thus bridges the gap between aspirational job applicants and industry expectations through the lens of intelligent automation and human-centric design

TABLE OF CONTENT

INTRODUCTION

1.1 Introduction

In the modern job market, recruiters and hiring managers are often faced with the daunting task of sifting through hundreds or even thousands of resumes to find the ideal candidate. This manual screening process is not only time-consuming but also prone to human bias and oversight. On the other side of the table, job seekers often struggle to align their resumes with the expectations of specific roles or industries due to a lack of timely feedback or personalized guidance. To address these challenges, the AI-Powered Resume Analyzer was conceived as an intelligent, automated solution that streamlines the resume evaluation process while empowering candidates with actionable insights.

This project harnesses the capabilities of artificial intelligence, natural language processing (NLP), and web technologies to create an interactive platform that analyzes resumes in real-time. Developed using Python and Streamlit, the application allows users to upload their resumes in PDF format, from which it extracts key information such as name, email, contact number, skills, number of pages, and more. These details are parsed using tools like pyresparser and pdfminer, ensuring accurate and comprehensive data extraction.

Once a resume is processed, the system performs a multidimensional evaluation. It identifies missing sections like Objective, Projects, Achievements, and Hobbies—elements that are critical in creating a strong first impression. Based on the content, a Resume Score is generated to give users a tangible measure of their resume's effectiveness. Moreover, the application classifies users based on their experience level (Fresher, Intermediate, or Experienced) by analyzing the number of pages in their resume.

A core innovation of this project lies in its intelligent Job Domain Prediction. By comparing the user's listed skills with predefined skill sets for different domains—such as Data Science, Web Development, Android Development, iOS Development, and UI/UX Design—the system predicts the most relevant job field for the user. It then recommends advanced or complementary skills to boost their competitiveness. To support further learning and upskilling, it also suggests relevant online courses and certifications, making the system not just analytical, but also developmental in nature.

Additionally, the system includes an Admin Dashboard which provides access to all analyzed user data and generates visual analytics in the form of pie charts. This feature can be particularly useful for educational institutions, training platforms, or recruitment agencies looking to gain insights into trends in candidate preparedness and interests.

In summary, the AI-Powered Resume Analyzer is more than just a resume screener—it's a career guidance platform. It combines the speed and precision of AI with the practicality of human-centric feedback, bridging the gap between job seekers and job opportunities in a smart, scalable, and insightful manner.

1.2 Problem Statement

In today's fast-paced and competitive job market, the volume of applications received by recruiters has increased dramatically, making manual resume screening an inefficient and error-prone task. Recruiters often face the challenge of shortlisting candidates quickly, which can result in potentially qualified candidates being overlooked due to non-standardized resumes, missing information, or lack of clarity in skill representation. This problem is particularly pressing in technical domains where skills must align closely with job requirements.

On the flip side, job seekers—especially fresh graduates and early-career professionals—often struggle to craft resumes that effectively communicate their strengths. They may be unaware of the specific skills and sections that make their resume stand out or lack insight into how to align their resumes with industry expectations. Additionally, they seldom receive meaningful feedback on their resume quality, making it difficult to identify what is missing or how to improve it. Moreover, traditional resume evaluation tools are often static, non-personalized, and limited in functionality. They rarely provide real-time recommendations for skills enhancement, domain suitability, or learning paths. There is a critical need for a smarter, interactive, and user-friendly system that not only evaluates resumes but also provides contextual feedback and personalized guidance.

The problem, therefore, is two-fold:

1. For Recruiters and Institutions: There is a lack of automated, intelligent tools to efficiently analyze and score resumes, classify candidates based on experience level, and gain insights into candidate suitability for specific job domains.
2. For Job Seekers: There is a lack of accessible, AI-driven tools that provide constructive feedback on resumes, help in identifying missing content, recommend relevant skills, and suggest appropriate learning resources based on individual backgrounds.

This project aims to bridge this gap by building an AI-Powered Resume Analyzer that performs intelligent parsing of resume content, predicts the most suitable career domain, evaluates resume quality, and offers actionable recommendations—all through a streamlined, web-based interface.

1.3 Objective

The primary objective of the AI-Powered Resume Analyzer project is to develop an intelligent, user-centric web application that automates the process of resume evaluation while offering personalized guidance to job seekers. The system is designed to accurately extract and analyze resume content using natural language processing techniques, identify key information such as skills, qualifications, and experience level, and provide constructive feedback to enhance resume quality. A major goal is to predict the most suitable job domain based on the user's listed skills and recommend relevant skills, courses, and certifications to improve employability. Additionally, the system aims to categorize candidates as Freshers, Intermediates, or Experienced professionals based on resume length, and generate a quantitative resume score that reflects the overall strength of their document. On the administrative side, the project seeks to create a backend dashboard that enables recruiters or educational institutions to access user data and visualize trends through interactive analytics. Ultimately, the objective is to empower both job seekers and recruiters by streamlining the resume review process, minimizing human bias, and providing actionable insights through an AI-driven platform.

1.4 Methodology

The methodology of the AI-Powered Resume Analyzer project is structured around the integration of natural language processing (NLP), machine learning techniques, and an interactive web interface to deliver a seamless and intelligent user experience. The process begins with resume upload, where users submit their resumes in PDF format through a Streamlit-based web application. The uploaded file is processed using the pdfminer library to extract raw text and pyresparser to parse essential details such as name, email, contact number, number of pages, and a list of identified skills. The system then classifies the user into one of three experience levels—Fresher, Intermediate, or Experienced—based on the number of resume pages.

Next, the extracted skills are compared against predefined keyword sets corresponding to five major job domains: Data Science, Web Development, Android Development, iOS Development, and UI/UX Design. This skill matching allows the system to predict the user's most likely field of interest or expertise. Once a domain is identified, the system dynamically generates recommendations for additional skills and curated online courses, enabling users to enhance their profile and bridge any skill gaps. The resume content is further analyzed for the presence of important sections such as Objectives, Projects, Achievements, and Hobbies, contributing to a computed resume score that reflects document completeness and professionalism.

All user data, including resume details, scores, predicted domain, and recommendations, is stored in a MySQL database for record-keeping and analytics. The admin panel, protected by login credentials, allows access to this data and provides visual summaries in the form of pie charts that show the distribution of candidate experience levels and preferred fields. This end-to-end methodology ensures a comprehensive and personalized resume evaluation process that benefits both applicants and reviewers through automation, feedback, and smart insights.

LITERATURE REVIEW

In recent years, the use of artificial intelligence in recruitment and resume analysis has gained significant traction due to the rising need for efficiency and objectivity in hiring processes. Several studies and systems have explored resume parsing using rule-based and machine learning methods. Traditional Applicant Tracking Systems (ATS), while widely used by recruiters, primarily rely on keyword matching and often lack contextual understanding, leading to inaccurate filtering and rejection of qualified candidates. Research in natural language processing (NLP) has advanced to enable deeper semantic analysis of unstructured text, allowing more accurate extraction of resume data such as skills, qualifications, and experiences. Tools like spaCy, NLTK, and resume parsers like pyresparser have made it easier to implement NLP pipelines for extracting structured information from resumes.

Various projects and papers have proposed using classification algorithms to match candidates to job roles based on their resume content. Some works have focused on skill-gap analysis and job recommendation engines by correlating resume data with job descriptions, but many of these systems fall short in delivering personalized feedback or improvement suggestions. Furthermore, most existing resume analysis tools are either commercially driven or restricted to enterprise use, leaving a gap in accessible and interactive platforms for individual users, especially students and freshers seeking guidance. The AI-Powered Resume Analyzer project builds on this research by integrating real-time resume parsing, domain prediction, resume scoring, and personalized skill and course recommendations into a unified platform. It fills a critical gap by not only evaluating resume quality but also guiding users toward career development using open-source technologies in a user-friendly web interface.

SYSTEM DEVELOPMENT

3.1 Analysis/Design/Development/Algorithm

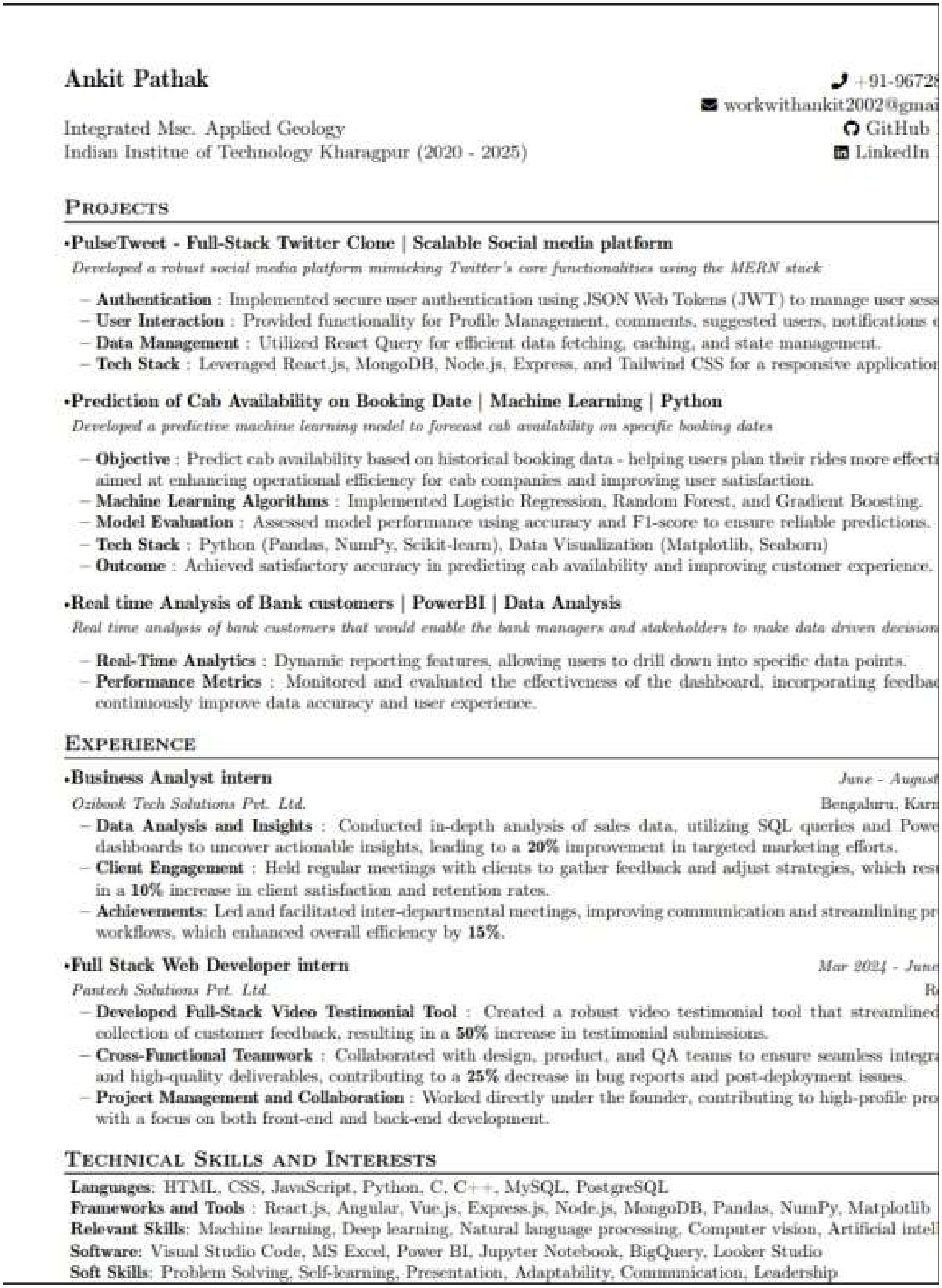
The development of the AI-Powered Resume Analyzer system followed a modular and iterative approach, integrating front-end, back-end, and data processing components into a cohesive web-based application. The system was built using Python as the core programming language, with Streamlit serving as the framework for creating an interactive and responsive user interface. For resume text extraction and parsing, libraries like pdfminer and pyresparser were employed. These tools enable the system to read PDF resumes and extract structured information such as the candidate's name, contact details, email, number of pages, and skills. Natural Language Processing libraries like NLTK and spaCy were also incorporated to improve text handling and keyword analysis.

The backend of the application uses MySQL for persistent storage of user data, including resume details, predicted career fields, recommended skills, resume scores, and suggested courses. To match user skills with potential job domains, predefined skill sets were manually curated for five major fields: Data Science, Web Development, Android Development, iOS Development, and UI/UX Design. Each of these categories contains a list of relevant technical and soft skills. Based on the skill match, the system intelligently predicts the most suitable field and recommends further skills and online courses to help the candidate improve their profile.

Additionally, the admin dashboard allows secure access to stored data and displays it using pie charts created with Plotly to show user trends and field preferences. This comprehensive system not only analyzes resumes but also stores data for further institutional insights and continuous improvement.

- Data Sources:
  - Resume files uploaded in PDF format by users.
  - Predefined keyword datasets for domain-specific skills (e.g., TensorFlow, React, Flutter, Swift, Adobe XD).
  - Curated course recommendations linked to platforms such as Coursera, Udemy, and edX.

- Technologies Used:
  - Python, Streamlit, MySQL, pdfminer, pyresparser, spaCy, NLTK, Plotly, PIL, and Streamlit Tags.Dataset contains three types of images:



Ankit Pathak

☎ +91-9672
✉ workwithankit2002@gmai
GitHub
LinkedIn

Integrated Msc. Applied Geology
Indian Institue of Technology Kharagpur (2020 - 2025)

Projects

•PulseTweet - Full-Stack Twitter Clone | Scalable Social media platform

Developed a robust social media platform mimicking Twitter's core functionalities using the MERN stack

- **Authentication** : Implemented secure user authentication using JSON Web Tokens (JWT) to manage user sess
- **User Interaction** : Provided functionality for Profile Management, comments, suggested users, notifications
- **Data Management** : Utilized React Query for efficient data fetching, caching, and state management.
- **Tech Stack** : Leveraged React.js, MongoDB, Node.js, Express, and Tailwind CSS for a responsive applicatio

•Prediction of Cab Availability on Booking Date | Machine Learning | Python

Developed a predictive machine learning model to forecast cab availability on specific booking dates

- **Objective** : Predict cab availability based on historical booking data - helping users plan their rides more effecti aimed at enhancing operational efficiency for cab companies and improving user satisfaction.
- **Machine Learning Algorithms** : Implemented Logistic Regression, Random Forest, and Gradient Boosting.
- **Model Evaluation** : Assessed model performance using accuracy and F1-score to ensure reliable predictions.
- **Tech Stack** : Python (Pandas, NumPy, Scikit-learn), Data Visualization (Matplotlib, Seaborn)
- **Outcome** : Achieved satisfactory accuracy in predicting cab availability and improving customer experience.

•Real time Analysis of Bank customers | PowerBI | Data Analysis

Real time analysis of bank customers that would enable the bank managers and stakeholders to make data driven decision

- **Real-Time Analytics** : Dynamic reporting features, allowing users to drill down into specific data points.
- **Performance Metrics** : Monitored and evaluated the effectiveness of the dashboard, incorporating feedba continuously improve data accuracy and user experience.

Experience

•Business Analyst intern *June - August*

Ozibook Tech Solutions Pvt. Ltd. Bengaluru, Kar

- **Data Analysis and Insights** : Conducted in-depth analysis of sales data, utilizing SQL queries and Powe dashboards to uncover actionable insights, leading to a **20%** improvement in targeted marketing efforts.
- **Client Engagement** : Held regular meetings with clients to gather feedback and adjust strategies, which res in a **10%** increase in client satisfaction and retention rates.
- **Achievements**: Led and facilitated inter-departmental meetings, improving communication and streamlining pr workflows, which enhanced overall efficiency by **15%**.

•Full Stack Web Developer intern *Mar 2024 - Jun*

Pantech Solutions Pvt. Ltd. R

- **Developed Full-Stack Video Testimonial Tool** : Created a robust video testimonial tool that streamlined collection of customer feedback, resulting in a **50%** increase in testimonial submissions.
- **Cross-Functional Teamwork** : Collaborated with design, product, and QA teams to ensure seamless integra and high-quality deliverables, contributing to a **25%** decrease in bug reports and post-deployment issues.
- **Project Management and Collaboration** : Worked directly under the founder, contributing to high-profile pro with a focus on both front-end and back-end development.

Technical Skills and Interests

Languages: HTML, CSS, JavaScript, Python, C, C++, MySQL, PostgreSQL
Frameworks and Tools : React.js, Angular, Vue.js, Express.js, Node.js, MongoDB, Pandas, NumPy, Matplotlib
Relevant Skills: Machine learning, Deep learning, Natural language processing, Computer vision, Artificial intel
Software: Visual Studio Code, MS Excel, Power BI, Jupyter Notebook, BigQuery, Looker Studio
Soft Skills: Problem Solving, Self-learning, Presentation, Adaptability, Communication, Leadership

3.2 Introduction to Natural Language Processing (NLP)

Natural Language Processing (NLP) is a subfield of Artificial Intelligence (AI) that focuses on enabling machines to understand, interpret, and generate human language. NLP involves a wide range of tasks, including language translation, sentiment analysis, named entity recognition (NER), speech recognition, text summarization, and text classification. One of the most common applications of NLP is text classification, which is the process of categorizing text into predefined categories or classes. Text classification plays a vital role in many applications, such as spam detection, sentiment analysis, and topic categorization.

In this context, text classification refers to the task of automatically assigning labels or categories to a given text document based on its content. Text classification is essential in the era of big data, where enormous amounts of text data are generated daily, such as from social media, news articles, and customer reviews. Efficient text classification allows businesses and researchers to derive meaningful insights from this vast amount of unstructured text data.

Key Concepts in Text Classification

Text classification involves several core steps, including preprocessing, feature extraction, and model training. Let’s explore these steps in more detail:

1. Text Preprocessing:

Text preprocessing is a crucial step in NLP that prepares the raw text data for further analysis. The goal of preprocessing is to clean the text, remove noise, and standardize it. Typical preprocessing techniques include:

- Tokenization: This process breaks text into smaller units such as words or phrases, which can then be analyzed individually.
- Lowercasing: Converting all text to lowercase to ensure that words like "Text" and "text" are treated the same.
- Removing Stop Words: Stop words like "and," "the," "is," and "in" do not provide significant meaning and are often removed.
- Stemming and Lemmatization: These processes reduce words to their base form (e.g., "running" becomes "run").
- Removing Special Characters and Punctuation: Unnecessary characters that do not contribute to the meaning of the text are discarded.

2. Feature Extraction:

Once the text is preprocessed, the next step is to convert it into numerical features that a machine learning model can understand. The most common feature extraction methods include:

- Bag of Words (BoW): This method involves representing each text document as a vector where each element corresponds to the frequency of a particular word in the document. While simple, BoW often suffers from high-dimensional feature space.
- TF-IDF (Term Frequency-Inverse Document Frequency): TF-IDF weighs the frequency of words in a document by their inverse frequency in the entire corpus. This technique helps identify important words in a document and mitigates the effect of common, less informative words.
- Word Embeddings (e.g., Word2Vec, GloVe): Word embeddings map words to continuous vector spaces, capturing semantic relationships between words. They help overcome the sparsity of BoW and the limitations of TF-IDF.

3. Model Training:

After feature extraction, the next step is to train a classification model on the labeled data. Several machine learning algorithms are commonly used for text classification tasks:

- Naive Bayes: This probabilistic classifier is based on Bayes' theorem and assumes independence between features. It is particularly useful for text classification problems like spam detection and sentiment analysis.
- Support Vector Machines (SVM): SVM is a supervised learning algorithm that finds the hyperplane that best separates the data into different classes. It is effective for text classification, especially in high-dimensional spaces like those created by text data.
- Logistic Regression: A linear model used for binary and multi-class classification tasks, logistic regression can be extended to text classification through the use of feature extraction methods such as BoW and TF-IDF.
- Deep Learning (Neural Networks): With the rise of deep learning, neural networks like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have become increasingly popular for text classification tasks. These models are capable of automatically learning hierarchical features from raw text data and often outperform traditional machine learning models.

4. Model Evaluation:

After training the model, it is essential to evaluate its performance using appropriate metrics. Common evaluation metrics for text classification include:

- Accuracy: The percentage of correctly classified instances.
- Precision: The fraction of true positive predictions among all positive predictions.
- Recall: The fraction of true positive predictions among all actual positive instances.
- F1-Score: The harmonic mean of precision and recall, useful when dealing with

imbalanced classes.

Cross-validation is often used to evaluate the generalization ability of the model, helping to prevent overfitting.

Applications of Text Classification

Text classification is used in a wide variety of applications across multiple domains. Some prominent examples include:

1. Spam Detection:

One of the most well-known applications of text classification is spam detection in emails. By classifying emails as "spam" or "ham" (non-spam), users can effectively filter unwanted content. Machine learning algorithms, such as Naive Bayes and SVM, are commonly used to build spam filters.

2. Sentiment Analysis:

Sentiment analysis involves classifying text into categories such as positive, negative, or neutral based on the sentiment expressed. This is widely used in social media monitoring, product reviews, and customer feedback analysis to gauge public opinion on various topics or products. For instance, a company might use sentiment analysis to assess customer sentiment toward a new product launch.

3. Topic Categorization:

In news articles, blogs, or research papers, topic categorization assigns texts to predefined categories based on their content, such as politics, sports, technology, health, etc. This helps in organizing large volumes of text data, making it easier for users to search and browse content.

4. Language Identification:

Language identification is the task of classifying text into different language categories. This can be useful for automatically detecting the language of a document or comment, enabling the use of appropriate language models and translation services.

5. Legal and Medical Text Classification:

In specialized fields like law and medicine, text classification is applied to categorize documents such as legal cases or medical reports. For example, a medical text classification model could classify patient records into various categories like "Diabetes," "Cardiology," or "Neurology."

Challenges in Text Classification

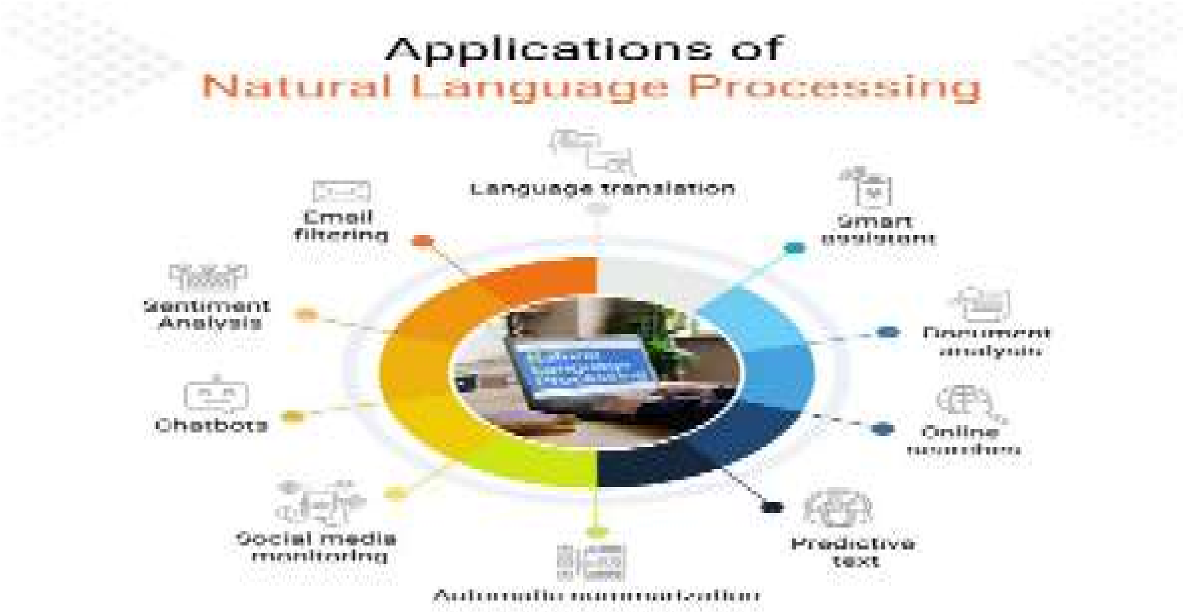
Despite its many successes, text classification presents several challenges:

- Ambiguity: Words and phrases can have different meanings depending on the context, making it difficult to classify text accurately.
- Data Imbalance: In many real-world datasets, some classes are underrepresented, leading to biased models.
- Multilingual Data: When dealing with text in multiple languages, handling diverse vocabularies and structures can complicate the classification process.
- Feature Selection: Deciding which features to include in the model is crucial, as irrelevant features can degrade model performance.

3.3 Overview of Natural Language Processing (NLP)

Natural Language Processing (NLP) is a branch of Artificial Intelligence (AI) that focuses on the interaction between computers and human languages. It enables machines to read, understand, interpret, and generate human language in a way that is both meaningful and useful. NLP bridges the gap between human communication and computer understanding, making it possible for computers to process, analyze, and even generate human language in various applications like translation, summarization, speech recognition, and text analysis.

NLP has become a vital technology in recent years due to the increasing need to process large amounts of unstructured text data. With the rapid growth of social media, online communication, and digital content, NLP is being used across industries for tasks such as sentiment analysis, automated translation, chatbots, and text classification.



Key Components of NLP

NLP is a multidisciplinary field that involves computer science, linguistics, and machine learning. It consists of several fundamental tasks that help in understanding and manipulating human language. Below are the key components of NLP:

1. Syntax and Parsing:

- Syntax refers to the grammatical structure of a language, determining how words and phrases are arranged to convey meaning.
- Parsing is the process of analyzing a sentence's structure to understand how different parts of speech relate to each other. For example, syntactic parsing helps to identify the subject, verb, and object in a sentence.
- Dependency Parsing: This technique focuses on identifying the dependencies between words in a sentence, such as subject-verb or object-verb relationships.

2. Semantics:

- Semantics involves understanding the meaning of words, phrases, and sentences. While syntax deals with structure, semantics deals with the meaning derived from the structure.
- Word Sense Disambiguation (WSD) is a common challenge in NLP, where the meaning of a word is context-dependent. For example, "bank" can refer to a financial institution or the side of a river, and WSD helps determine the correct meaning based on context.

3. Morphology:

- Morphology is the study of the structure of words and their components (morphemes). A morpheme is the smallest unit of meaning in a language. For example, in the word "unhappiness," "un-" is a prefix, "happy" is the root, and "-ness" is a suffix.

Understanding word morphology helps in tasks like stemming and lemmatization, which simplify words to their base forms (e.g., "running" becomes "run").

4. Pragmatics:

- Pragmatics deals with understanding how context affects the interpretation of language. It goes beyond literal meanings and takes into account factors like tone, intent, and conversational context.
- For example, "Can you pass the salt?" is a request, not a question about someone's ability to pass the salt. Pragmatics helps in identifying such nuances in language.

5. Discourse:

- Discourse refers to the larger context of language, where meaning is derived from entire conversations or paragraphs rather than individual sentences.
- NLP models use discourse analysis to understand how sentences or phrases in a text relate to one another to form coherent meaning over larger chunks of text.

6. Speech Recognition:

- Speech recognition is an area of NLP where spoken language is converted into written text. This technology is used in applications like virtual assistants (e.g., Siri, Alexa) and transcription services.
- It involves understanding speech patterns, accents, and noisy environments to accurately convert speech to text.

NLP Tasks and Applications

NLP encompasses a wide range of tasks, each focusing on different aspects of language. Some common NLP tasks and their applications include:

1. Text Classification:

- Text classification involves categorizing text into predefined categories or classes. This is used in applications such as spam detection, sentiment analysis, and topic categorization.

2. Named Entity Recognition (NER):

- NER is the task of identifying and classifying named entities (such as names of people, organizations, locations, dates, etc.) in a text. It is commonly used in information extraction, where important data needs to be extracted from unstructured text.

3. Machine Translation:

- Machine translation is the automatic translation of text or speech from one language to another. Google Translate and DeepL are examples of popular machine translation systems

4. Sentiment Analysis:

- Sentiment analysis involves analyzing a text to determine its sentiment—whether the writer's opinion is positive, negative, or neutral. It is widely used in social media monitoring, brand reputation management, and customer feedback analysis

5. Text Summarization:

- Text summarization is the task of generating a shortened version of a text while preserving its key information. This is useful for news aggregation, document summarization, and content curation.

6. Question Answering:

- In question answering systems, the goal is to generate precise answers to user queries from a large corpus of text. Systems like Google Search, Siri, and Alexa employ NLP techniques to understand questions and provide relevant answers.

7. Chatbots and Conversational Agents:

- Chatbots use NLP to engage in real-time conversations with users, offering services like customer support, product recommendations, and troubleshooting. These systems rely on NLP to process user input and generate appropriate responses.

8. Speech-to-Text (STT) and Text-to-Speech (TTS):

- Speech-to-text (STT) converts spoken language into written text, while text-to-speech (TTS) converts written text into spoken language. These technologies are used in virtual assistants, transcription services, and accessibility applications.

9. Language Generation:

- Language generation focuses on creating human-like text from structured data or input. This is used in applications like automatic report generation, content creation, and even creative writing.

Approaches in NLP

There are several approaches to solving NLP problems, ranging from traditional rule-based systems to modern machine learning and deep learning techniques.

1. Rule-based Approaches:

- In the early days of NLP, systems relied heavily on hand-crafted rules for parsing, grammar, and language understanding. While effective for specific tasks, rule-based systems are limited in scalability and flexibility.

2. Statistical and Machine Learning Approaches:

- As data became more abundant, statistical and machine learning methods gained popularity. Techniques like decision trees, support vector machines (SVM), and Naive Bayes are used for tasks such as classification and regression.
- Feature extraction methods, such as TF-IDF, Bag of Words, and word embeddings, are used to represent text data numerically.

3. Deep Learning Approaches:

- Deep learning has revolutionized NLP in recent years, particularly with the advent of models like Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and Transformers (e.g., BERT, GPT).
- These models can learn complex representations of language and have dramatically improved performance in tasks like machine translation, text generation, and question answering.

Challenges in NLP

Despite significant advances, NLP still faces many challenges:

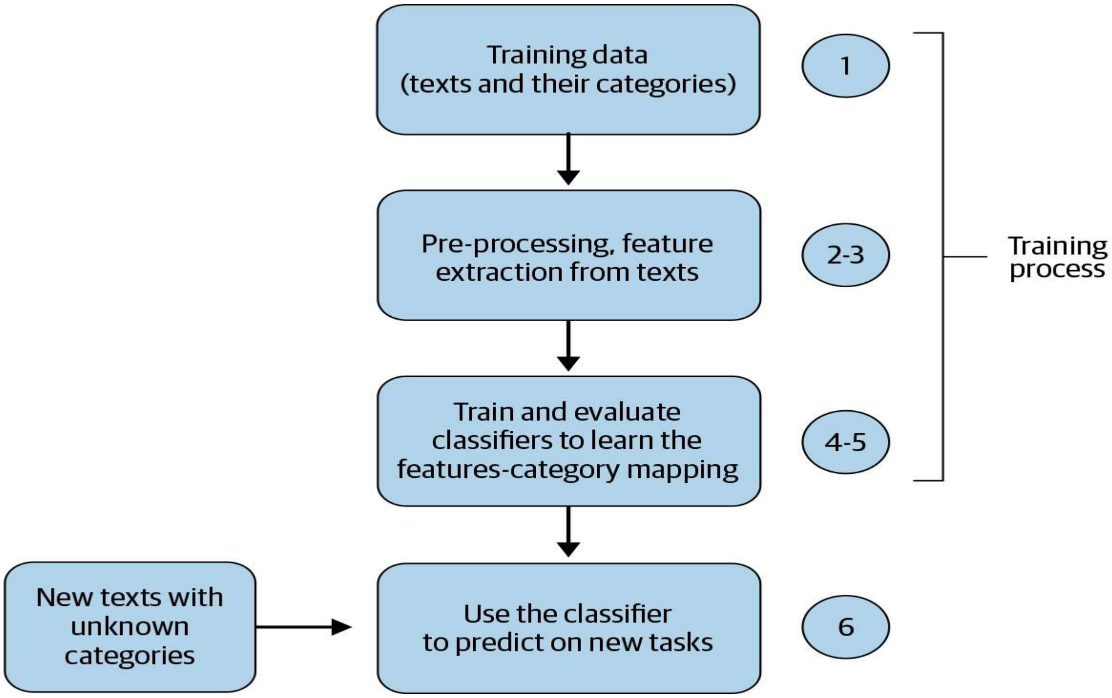
- Ambiguity: Words and phrases can have multiple meanings depending on context, making interpretation difficult for machines.

- Contextual Understanding: Understanding the context of language, such as sarcasm, humor, or cultural nuances, remains a challenge for NLP systems.

- Language Diversity: There are thousands of languages, dialects, and variations in the world, making it difficult to develop universal NLP models that work across all languages.

- Data Sparsity: While large datasets are available for widely spoken languages, many languages and specialized domains lack sufficient data for training accurate models.

3.3 Model Development in NLP for Text Classification


Training data
(texts and their categories)
1
Pre-processing, feature extraction from texts
2-3
Training process
Train and evaluate classifiers to learn the features-category mapping
4-5
New texts with unknown categories
Use the classifier to predict on new tasks
6

Text classification is one of the foundational tasks in Natural Language Processing (NLP). It involves categorizing text into predefined labels or categories based on its content. Text classification is used in numerous applications such as sentiment analysis, spam detection, topic categorization, and news classification. The development of models for text classification typically follows a series of stages, from data preprocessing to model evaluation and optimization. This process involves choosing the appropriate techniques, algorithms, and tools for building effective and accurate models.

1. Data Collection and Preprocessing

The first step in developing a text classification model is data collection. A substantial amount of labeled text data is needed to train the model. Depending on the task, this data could come from various sources such as news articles, customer reviews, social media posts, or research papers. The quality and quantity of data play a crucial role in the performance of the model.

Once data is collected, preprocessing is necessary to transform raw text into a format suitable for machine learning algorithms. The key steps involved in text preprocessing include:

- Text Cleaning: This step involves removing irrelevant elements from the text, such as special characters, punctuation, HTML tags, or any non-alphabetic characters that may not contribute to the model's understanding.
- Tokenization: Tokenization is the process of breaking down text into smaller units, usually words or subwords. For example, the sentence "I love programming" would be tokenized into ["I", "love", "programming"].
- Stop-word Removal: Stop words are common words (like "the", "and", "is") that do not carry significant meaning in the context of text classification and are usually removed to reduce noise in the data.
- Lowercasing: Converting all text to lowercase helps standardize the data and avoid treating the same word in different cases as distinct.
- Stemming and Lemmatization: These techniques are used to reduce words to their root forms. For instance, "running" can be reduced to "run" using stemming or lemmatization.
- Text Vectorization: Since machine learning models cannot directly work with raw text, text data needs to be converted into numerical format. This can be achieved through techniques like:

- Bag of Words (BoW): A simple representation where each word in the document is assigned a unique index, and the document is represented by a vector containing the frequency of each word.
- TF-IDF (Term Frequency-Inverse Document Frequency): This method not only counts the frequency of words but also weighs words based on their importance in the corpus. Words that occur frequently in a document but rarely across other documents are given higher weights.
- Word Embeddings: In modern NLP, more sophisticated representations like word embeddings (e.g., Word2Vec, GloVe) capture semantic relationships between words by mapping them into continuous vector spaces. Each word is represented as a dense vector that encodes its meaning based on context.

2. Model Selection

Once the data is preprocessed and represented numerically, the next step is choosing the appropriate machine learning model for the text classification task. The choice of model depends on the complexity of the task and the characteristics of the dataset. Several popular models are commonly used in text classification:

- Logistic Regression: A simple but powerful linear model often used as a baseline in binary classification tasks. It works well when the relationship between features (words) and the target (labels) is linear.
- Naive Bayes Classifier: A probabilistic model based on Bayes' theorem. It assumes independence between features and works well for text classification, especially when features (words) are conditionally independent given the class label.
- Support Vector Machine (SVM): A supervised learning algorithm that finds the hyperplane that best separates different classes. SVM is effective in high-dimensional spaces, which is typical in text classification problems.
- Deep Learning Models:
  - Recurrent Neural Networks (RNNs): RNNs are particularly useful for tasks

involving sequences, as they have a memory mechanism that captures contextual information across words in a sentence. For text classification, RNNs are used to capture the temporal dependency between words in a document.

- Long Short-Term Memory (LSTM): LSTMs are a type of RNN that addresses the vanishing gradient problem, enabling them to capture long-range dependencies in text. LSTM-based models are effective in complex text classification tasks.
- Convolutional Neural Networks (CNNs): While CNNs are typically associated with image processing, they have shown promising results in text classification tasks. By applying convolutional filters over text, CNNs can capture local patterns like n-grams and extract hierarchical features from the text.
- Transformers (BERT, GPT, etc.): Transformer models, particularly BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pretrained Transformer), have revolutionized NLP tasks. BERT is pre-trained on a large corpus and can be fine-tuned for specific tasks such as text classification. These models capture contextual relationships between words more effectively than traditional methods.

3. Model Training

Once the model is selected, the next step is training the model on the labeled dataset. The training process involves optimizing the model parameters to minimize the loss function, which measures the error between the model's predictions and the true labels.

- Hyperparameter Tuning: This step involves adjusting the model’s hyperparameters, such as the learning rate, regularization strength, and number of layers in neural networks, to improve performance. This can be done using techniques like grid search, random search, or more advanced methods like Bayesian optimization.
- Cross-Validation: To ensure that the model generalizes well to unseen data, k-fold cross-validation is often used. This involves splitting the training data into k subsets

and training the model k times, each time with a different subset held out for validation.

4. Model Evaluation

After the model is trained, it needs to be evaluated on a separate test set to assess its performance. Common evaluation metrics for text classification include:

- Accuracy: The proportion of correctly classified instances to the total number of instances. While simple, accuracy can be misleading if the dataset is imbalanced (e.g., if one class is much larger than the other).
- Precision and Recall: Precision measures the proportion of true positive predictions out of all positive predictions, while recall measures the proportion of true positive predictions out of all actual positives. These metrics are important when dealing with imbalanced classes.
- F1-Score: The harmonic mean of precision and recall, providing a balanced measure between the two metrics.
- ROC-AUC: The area under the Receiver Operating Characteristic curve is used to evaluate the trade-off between true positive rate and false positive rate at different thresholds.

5. Model Optimization and Deployment

Once the model is evaluated, further improvements may be made. This could include:

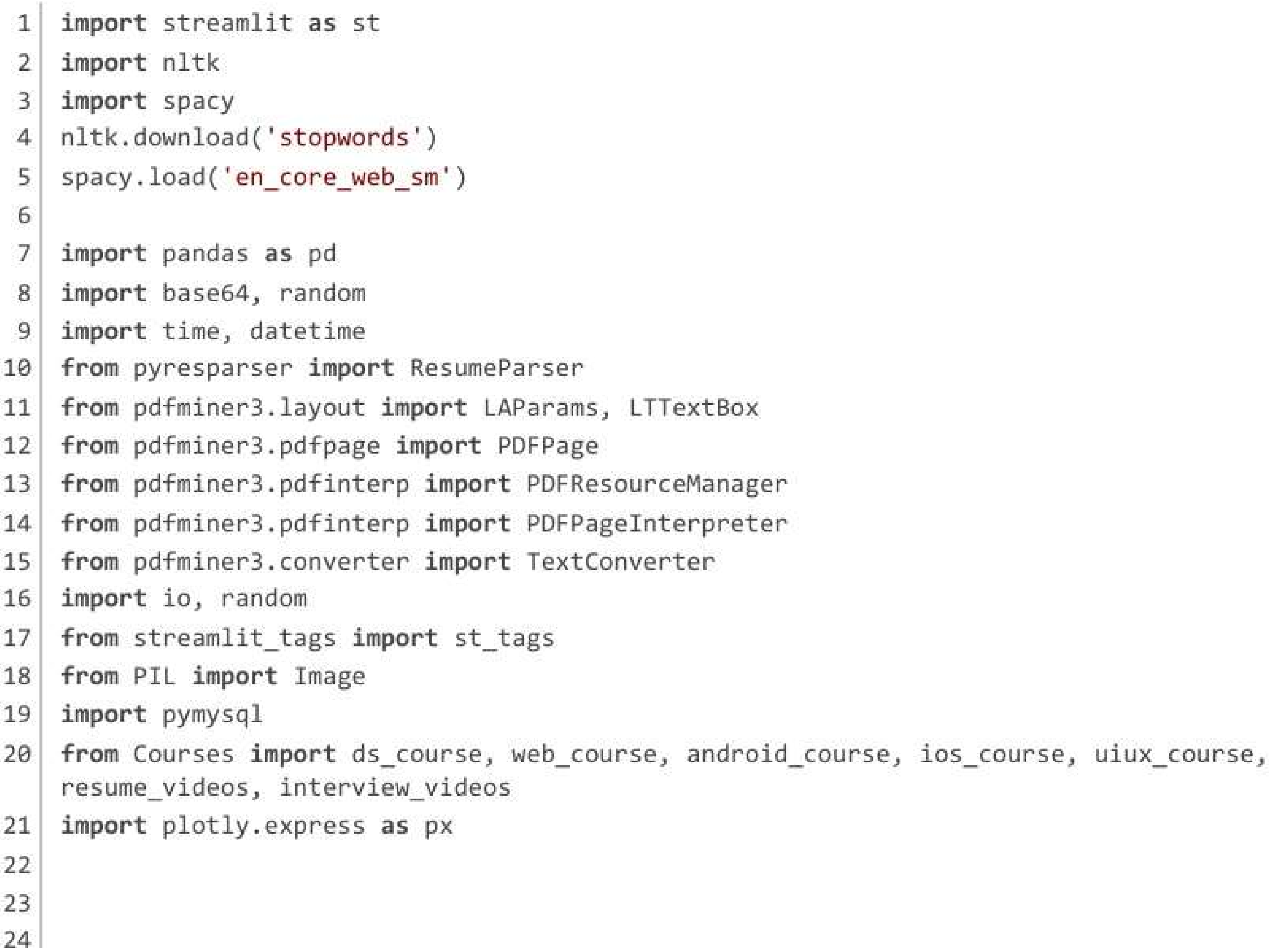
- Model Fine-Tuning: Fine-tuning the model by retraining it on additional data or adjusting the learning rate and other hyperparameters.
- Ensemble Methods: Combining multiple models (e.g., through techniques like boosting, bagging, or stacking) can improve classification performance by leveraging the strengths of different models.

Once the model achieves satisfactory performance, it can be deployed in real-world applications such as email spam detection, content moderation, sentiment analysis, or news categorizationon testing set. Then model is saved and deployed over an

After collecting the data, first step is data cleaning and pre-processing for which I am using tf dataset and data augmentation. Data augmentation used when dataset is not that large then we can just rotate or increase decrease the contrast of the image to create a new image which can be used as an input. Next step is model building using convolutional neural network. CNN is a standard way for image classification as it provides better accuracy as compare to normal neural network. Then saving the model for deployment. Then backend of the web application is designed by FastApi. Below is the figure showing state transition diagram.

3.4 Analytical

First of all, importing all the dependencies

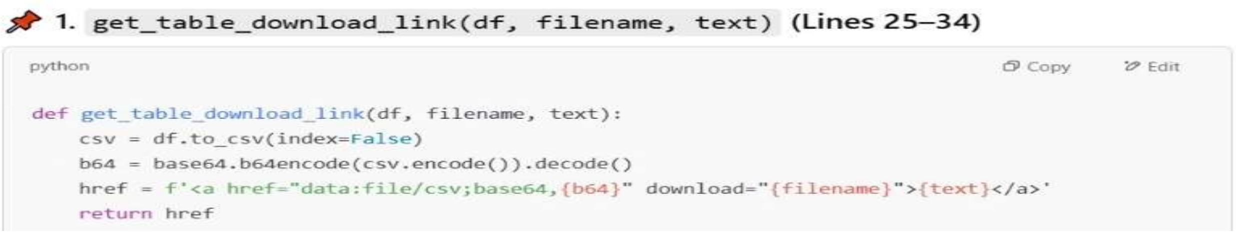


```python
import streamlit as st
import nltk
import spacy
nltk.download('stopwords')
spacy.load('en_core_web_sm')

import pandas as pd
import base64, random
import time, datetime
from pyresparser import ResumeParser
from pdfminer3.layout import LAParams, LTTextBox
from pdfminer3.pdfpage import PDFPage
from pdfminer3.pdfinterp import PDFResourceManager
from pdfminer3.pdfinterp import PDFPageInterpreter
from pdfminer3.converter import TextConverter
import io, random
from streamlit_tags import st_tags
from PIL import Image
import pymysql
from Courses import ds_course, web_course, android_course, ios_course, uiux_course, resume_videos, interview_videos
import plotly.express as px
```

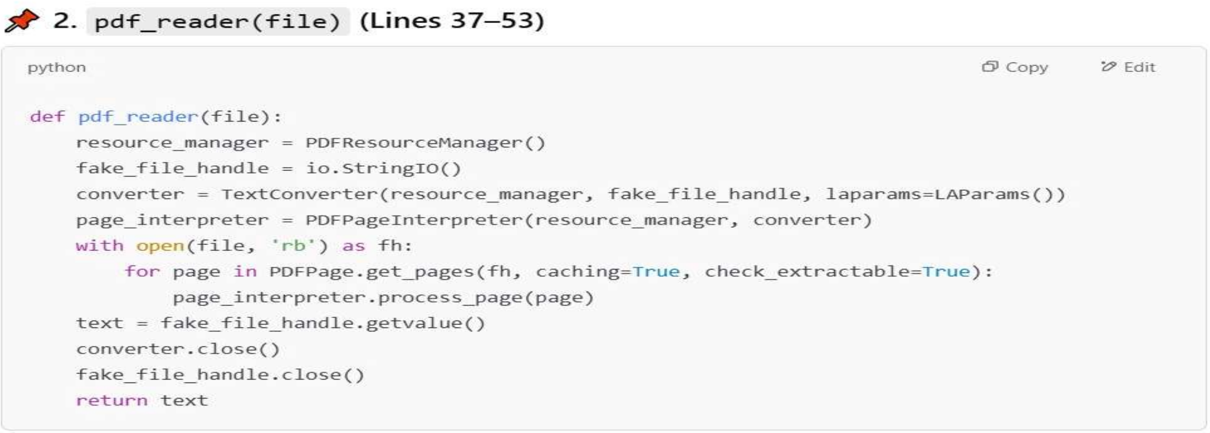
Importing Required Libraries

This initial segment of the project sets up the core environment by importing necessary libraries and modules. Each library serves a specific role in the overall functioning of the resume analyzer app.



📌 1. `get_table_download_link(df, filename, text)` (Lines 25–34)

```python
def get_table_download_link(df, filename, text):
    csv = df.to_csv(index=False)
    b64 = base64.b64encode(csv.encode()).decode()
    href = f'<a href="data:file/csv;base64,{b64}" download="{filename}">{text}</a>'
    return href
```

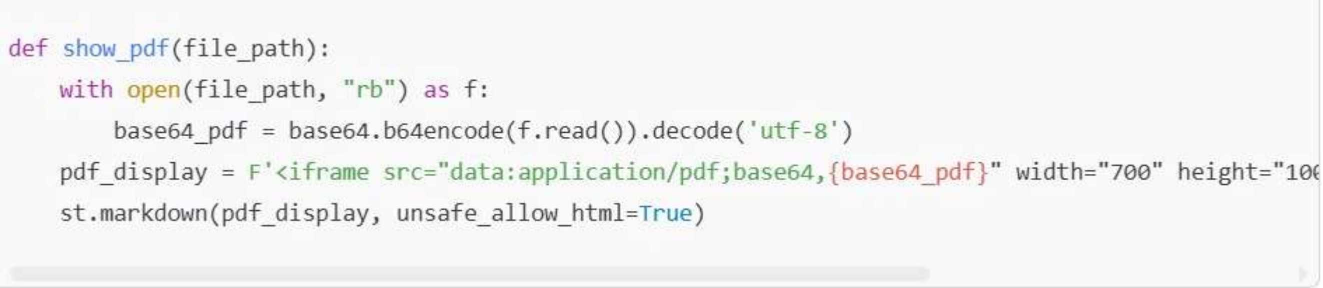


📌 2. `pdf_reader(file)` (Lines 37–53)

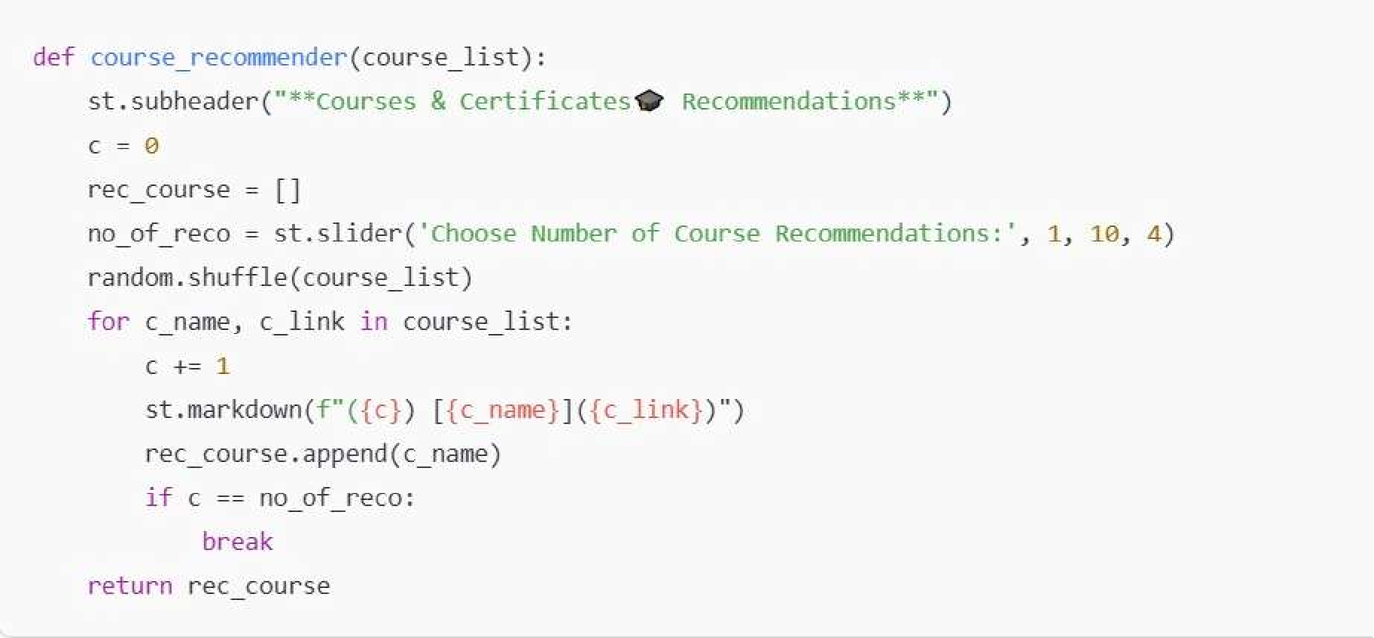
```python
def pdf_reader(file):
    resource_manager = PDFResourceManager()
    fake_file_handle = io.StringIO()
    converter = TextConverter(resource_manager, fake_file_handle, laparams=LAParams())
    page_interpreter = PDFPageInterpreter(resource_manager, converter)
    with open(file, 'rb') as fh:
        for page in PDFPage.get_pages(fh, caching=True, check_extractable=True):
            page_interpreter.process_page(page)
    text = fake_file_handle.getvalue()
    converter.close()
    fake_file_handle.close()
    return text
```

Utility Functions

This segment includes four helper functions that enhance the usability and modularity of the project. These utilities cover everything from file downloads to PDF reading and course recommendation.



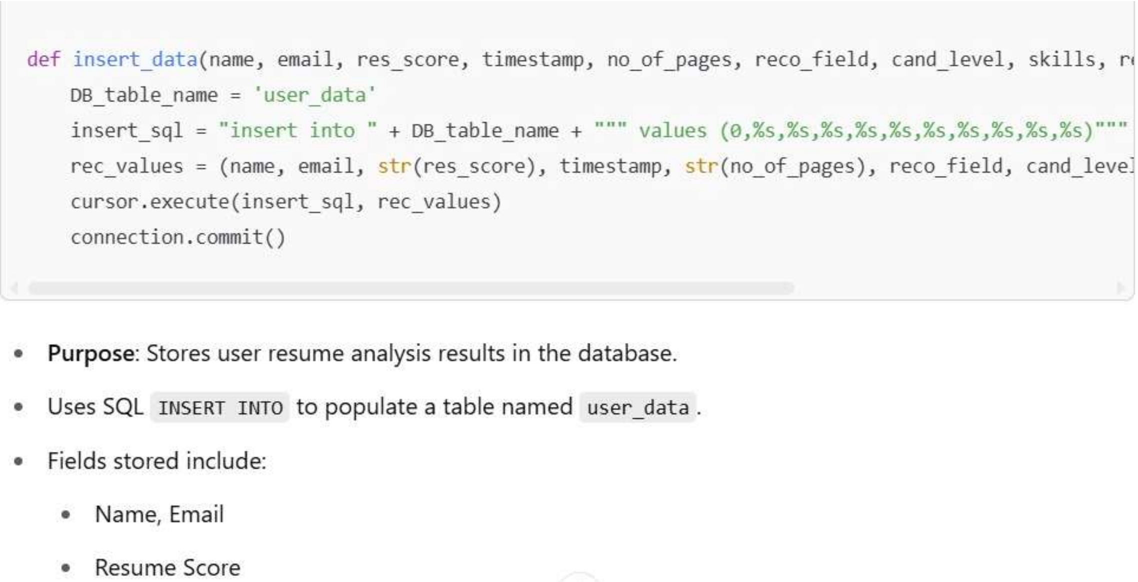
```
def show_pdf(file_path):
    with open(file_path, "rb") as f:
        base64_pdf = base64.b64encode(f.read()).decode('utf-8')
    pdf_display = F'<iframe src="data:application/pdf;base64,{base64_pdf}" width="700" height="10
    st.markdown(pdf_display, unsafe_allow_html=True)
```



```
def course_recommender(course_list):
    st.subheader("**Courses & Certificates🎓 Recommendations**")
    c = 0
    rec_course = []
    no_of_reco = st.slider('Choose Number of Course Recommendations:', 1, 10, 4)
    random.shuffle(course_list)
    for c_name, c_link in course_list:
        c += 1
        st.markdown(f"({c}) [{c_name}]({c_link})")
        rec_course.append(c_name)
        if c == no_of_reco:
            break
    return rec_course
```

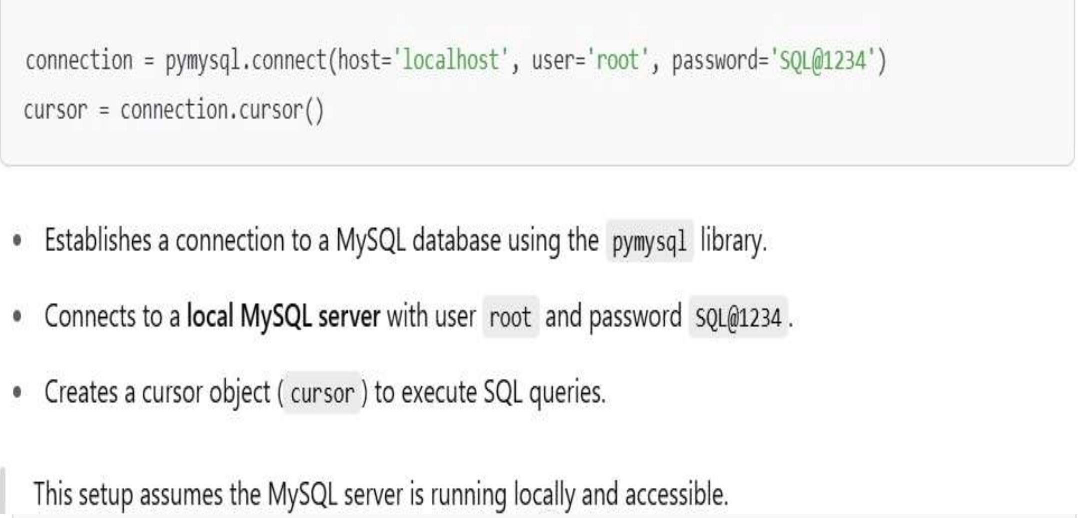
Database Configuration and Data Insertion

This segment handles everything related to database connectivity, table creation, and storing analyzed resume data. It ensures all user information and recommendations are persistently stored for future access, especially for the admin dashboard.



```python
def insert_data(name, email, res_score, timestamp, no_of_pages, reco_field, cand_level, skills, r
    DB_table_name = 'user_data'
    insert_sql = "insert into " + DB_table_name + """ values (0,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"""
    rec_values = (name, email, str(res_score), timestamp, str(no_of_pages), reco_field, cand_level
    cursor.execute(insert_sql, rec_values)
    connection.commit()
```

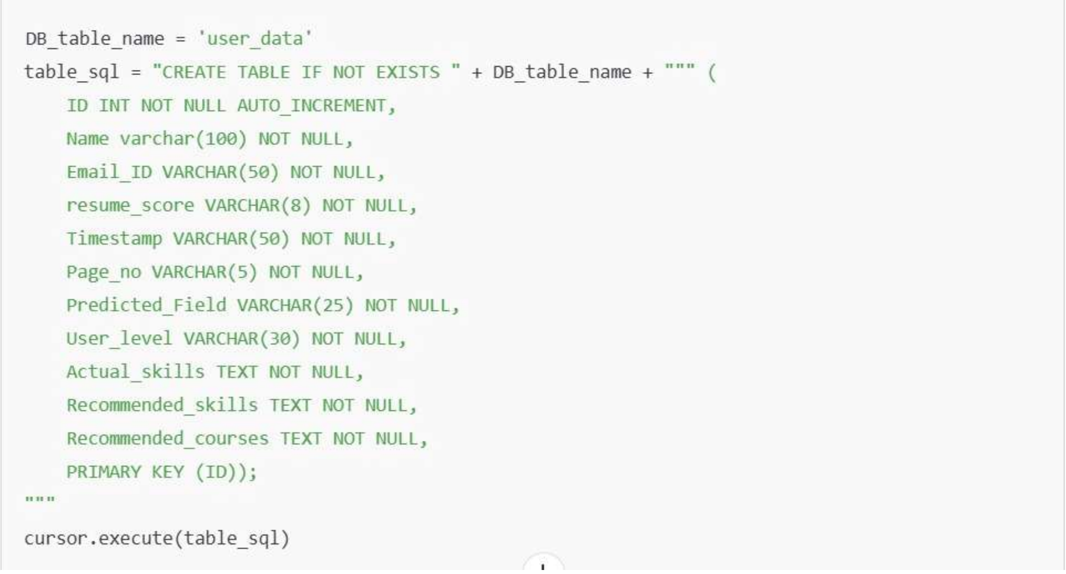
- **Purpose**: Stores user resume analysis results in the database.
- Uses SQL `INSERT INTO` to populate a table named `user_data`.
- Fields stored include:
  - Name, Email
  - Resume Score



```python
connection = pymysql.connect(host='localhost', user='root', password='SQL@1234')
cursor = connection.cursor()
```

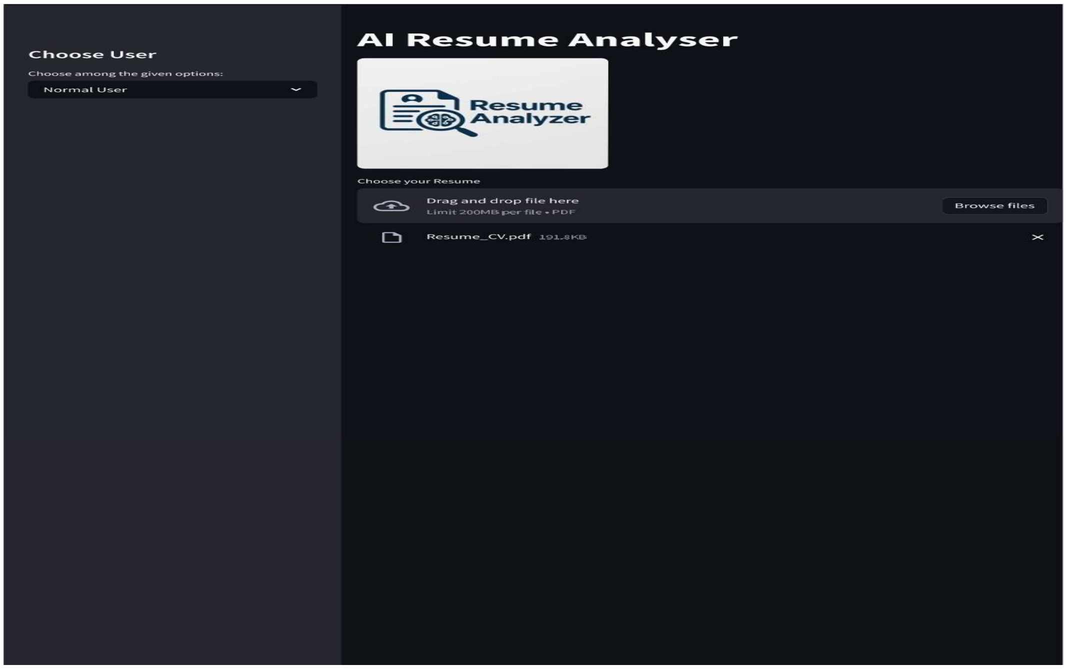
- Establishes a connection to a MySQL database using the `pymysql` library.
- Connects to a **local MySQL server** with user `root` and password `SQL@1234`.
- Creates a cursor object (`cursor`) to execute SQL queries.

> This setup assumes the MySQL server is running locally and accessible.



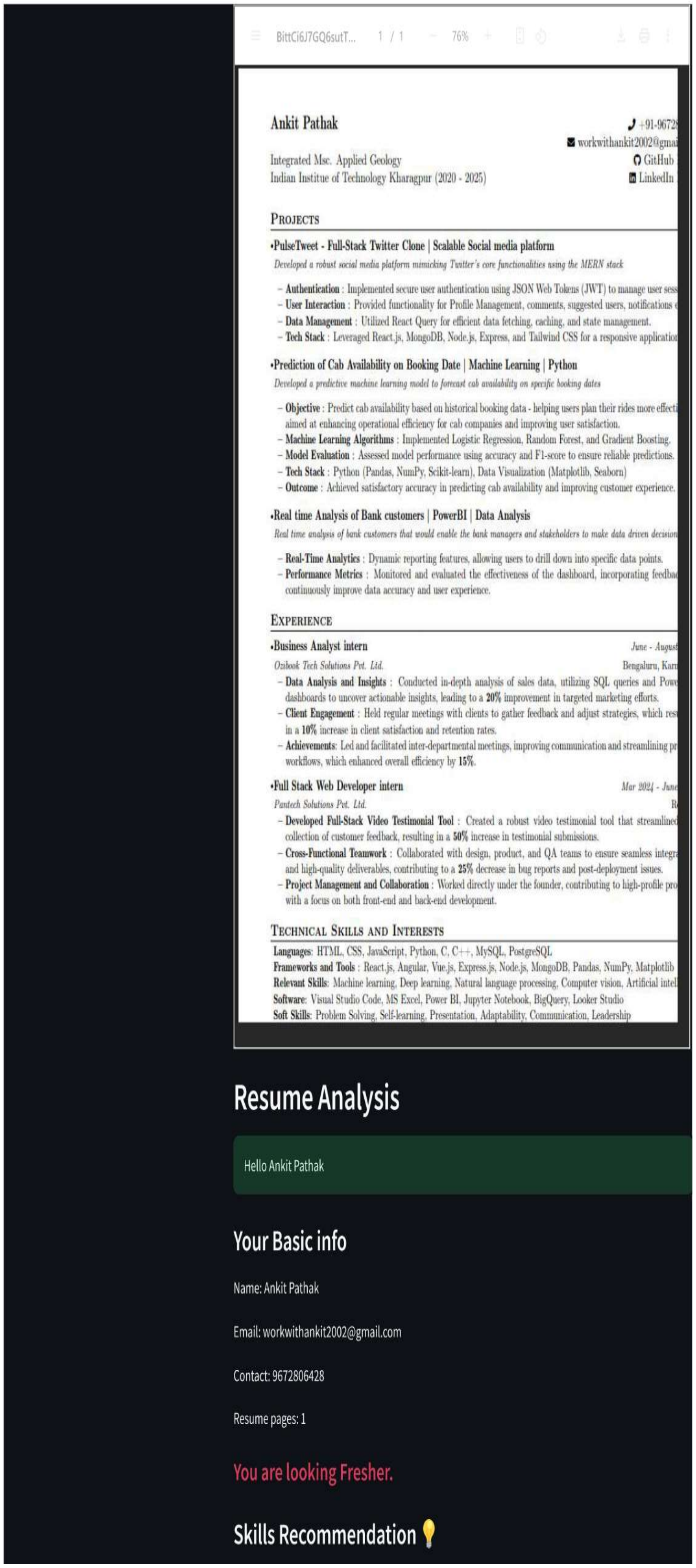
```python
DB_table_name = 'user_data'
table_sql = "CREATE TABLE IF NOT EXISTS " + DB_table_name + """ (
    ID INT NOT NULL AUTO_INCREMENT,
    Name varchar(100) NOT NULL,
    Email_ID VARCHAR(50) NOT NULL,
    resume_score VARCHAR(8) NOT NULL,
    Timestamp VARCHAR(50) NOT NULL,
    Page_no VARCHAR(5) NOT NULL,
    Predicted_Field VARCHAR(25) NOT NULL,
    User_level VARCHAR(30) NOT NULL,
    Actual_skills TEXT NOT NULL,
    Recommended_skills TEXT NOT NULL,
    Recommended_courses TEXT NOT NULL,
    PRIMARY KEY (ID));
"""
cursor.execute(table_sql)
```

This segment is responsible for configuring the Streamlit web app interface, initializing the app structure, and letting users choose between the Normal User and Admin workflows Streamlit Page Configuration



Choose among the given options: Normal User

Resume_CV.pdf 191.8KB



BittCi6J7GQ6sutT... 1 / 1 76%

Ankit Pathak

+91-96728
workwithankit2002@gmai
GitHub
LinkedIn

Integrated Msc. Applied Geology
Indian Institue of Technology Kharagpur (2020 - 2025)

PROJECTS

•PulseTweet - Full-Stack Twitter Clone | Scalable Social media platform

Developed a robust social media platform mimicking Twitter's core functionalities using the MERN stack

- **Authentication** : Implemented secure user authentication using JSON Web Tokens (JWT) to manage user sess
- **User Interaction** : Provided functionality for Profile Management, comments, suggested users, notifications
- **Data Management** : Utilized React Query for efficient data fetching, caching, and state management.
- **Tech Stack** : Leveraged React.js, MongoDB, Node.js, Express, and Tailwind CSS for a responsive applicatio

•Prediction of Cab Availability on Booking Date | Machine Learning | Python

Developed a predictive machine learning model to forecast cab availability on specific booking dates

- **Objective** : Predict cab availability based on historical booking data - helping users plan their rides more effecti aimed at enhancing operational efficiency for cab companies and improving user satisfaction.
- **Machine Learning Algorithms** : Implemented Logistic Regression, Random Forest, and Gradient Boosting.
- **Model Evaluation** : Assessed model performance using accuracy and F1-score to ensure reliable predictions.
- **Tech Stack** : Python (Pandas, NumPy, Scikit-learn), Data Visualization (Matplotlib, Seaborn)
- **Outcome** : Achieved satisfactory accuracy in predicting cab availability and improving customer experience.

•Real time Analysis of Bank customers | PowerBI | Data Analysis

Real time analysis of bank customers that would enable the bank managers and stakeholders to make data driven decision

- **Real-Time Analytics** : Dynamic reporting features, allowing users to drill down into specific data points.
- **Performance Metrics** : Monitored and evaluated the effectiveness of the dashboard, incorporating feedba continuously improve data accuracy and user experience.

EXPERIENCE

•Business Analyst intern *June - August*

Ozibook Tech Solutions Pvt. Ltd. Bengaluru, Karn

- **Data Analysis and Insights** : Conducted in-depth analysis of sales data, utilizing SQL queries and Powe dashboards to uncover actionable insights, leading to a **20%** improvement in targeted marketing efforts.
- **Client Engagement** : Held regular meetings with clients to gather feedback and adjust strategies, which res in a **10%** increase in client satisfaction and retention rates.
- **Achievements**: Led and facilitated inter-departmental meetings, improving communication and streamlining pr workflows, which enhanced overall efficiency by **15%**.

•Full Stack Web Developer intern *Mar 2024 - June*

Pantech Solutions Pvt. Ltd. R

- **Developed Full-Stack Video Testimonial Tool** : Created a robust video testimonial tool that streamlined collection of customer feedback, resulting in a **50%** increase in testimonial submissions.
- **Cross-Functional Teamwork** : Collaborated with design, product, and QA teams to ensure seamless integra and high-quality deliverables, contributing to a **25%** decrease in bug reports and post-deployment issues.
- **Project Management and Collaboration** : Worked directly under the founder, contributing to high-profile pro with a focus on both front-end and back-end development.

TECHNICAL SKILLS AND INTERESTS

Languages: HTML, CSS, JavaScript, Python, C, C++, MySQL, PostgreSQL
Frameworks and Tools : React.js, Angular, Vue.js, Express.js, Node.js, MongoDB, Pandas, NumPy, Matplotlib
Relevant Skills: Machine learning, Deep learning, Natural language processing, Computer vision, Artificial intell
Software: Visual Studio Code, MS Excel, Power BI, Jupyter Notebook, BigQuery, Looker Studio
Soft Skills: Problem Solving, Self-learning, Presentation, Adaptability, Communication, Leadership

Resume Analysis

Hello Ankit Pathak

Your Basic info

Name: Ankit Pathak

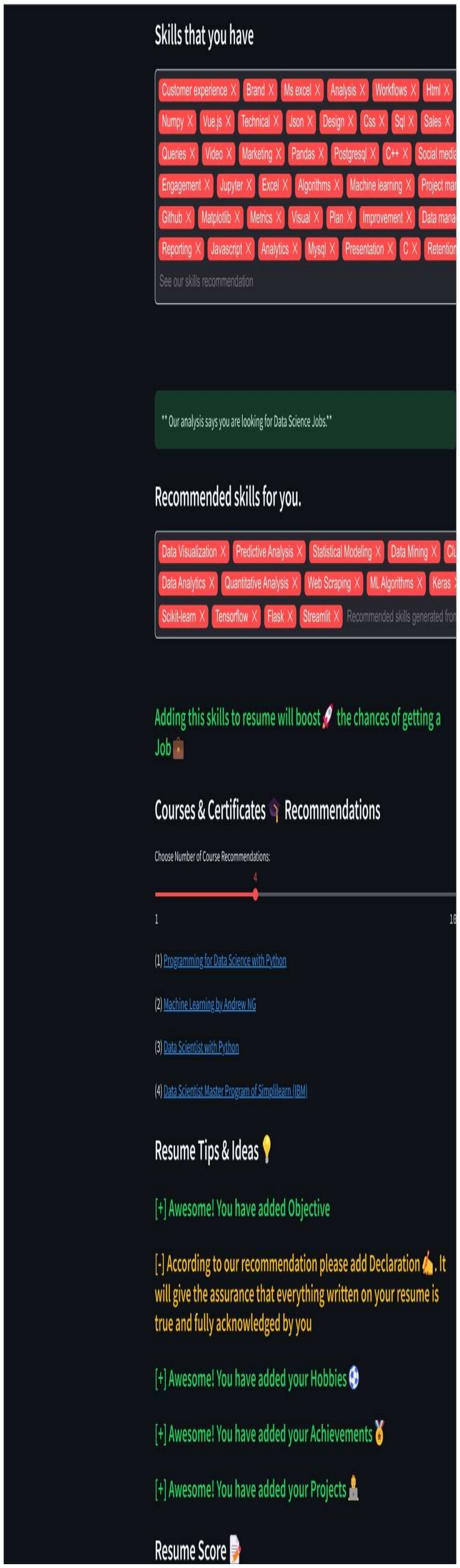
Email: workwithankit2002@gmail.com

Contact: 9672806428

Resume pages: 1

You are looking Fresher.

Skills Recommendation 💡



Skills that you have

Customer experience X Brand X Ms excel X Analysis X Workflows X Html X
Numpy X Vue.js X Technical X Json X Design X Css X Sql X Sales X
Queries X Video X Marketing X Pandas X Postgresql X C++ X Social media
Engagement X Jupyter X Excel X Algorithms X Machine learning X Project mar
Github X Matplotlib X Metrics X Visual X Plan X Improvement X Data mana
Reporting X Javascript X Analytics X Mysql X Presentation X C X Retention

See our skills recommendation

** Our analysis says you are looking for Data Science Jobs.**

Recommended skills for you.

Data Visualization X Predictive Analysis X Statistical Modeling X Data Mining X Clu
Data Analytics X Quantitative Analysis X Web Scraping X ML Algorithms X Keras X
Scikit-learn X Tensorflow X Flask X Streamlit X Recommended skills generated fron

Adding this skills to resume will boost 🚀 the chances of getting a Job 💼

Courses & Certificates 🎓 Recommendations

Choose Number of Course Recommendations:

(1) Programming for Data Science with Python

(2) Machine Learning by Andrew NG

(3) Data Scientist with Python

(4) Data Scientist Master Program of Simplilearn (IBM)

Resume Tips & Ideas 💡

[+] Awesome! You have added Objective

[-] According to our recommendation please add Declaration ✍. It will give the assurance that everything written on your resume is true and fully acknowledged by you

[+] Awesome! You have added your Hobbies ⚽

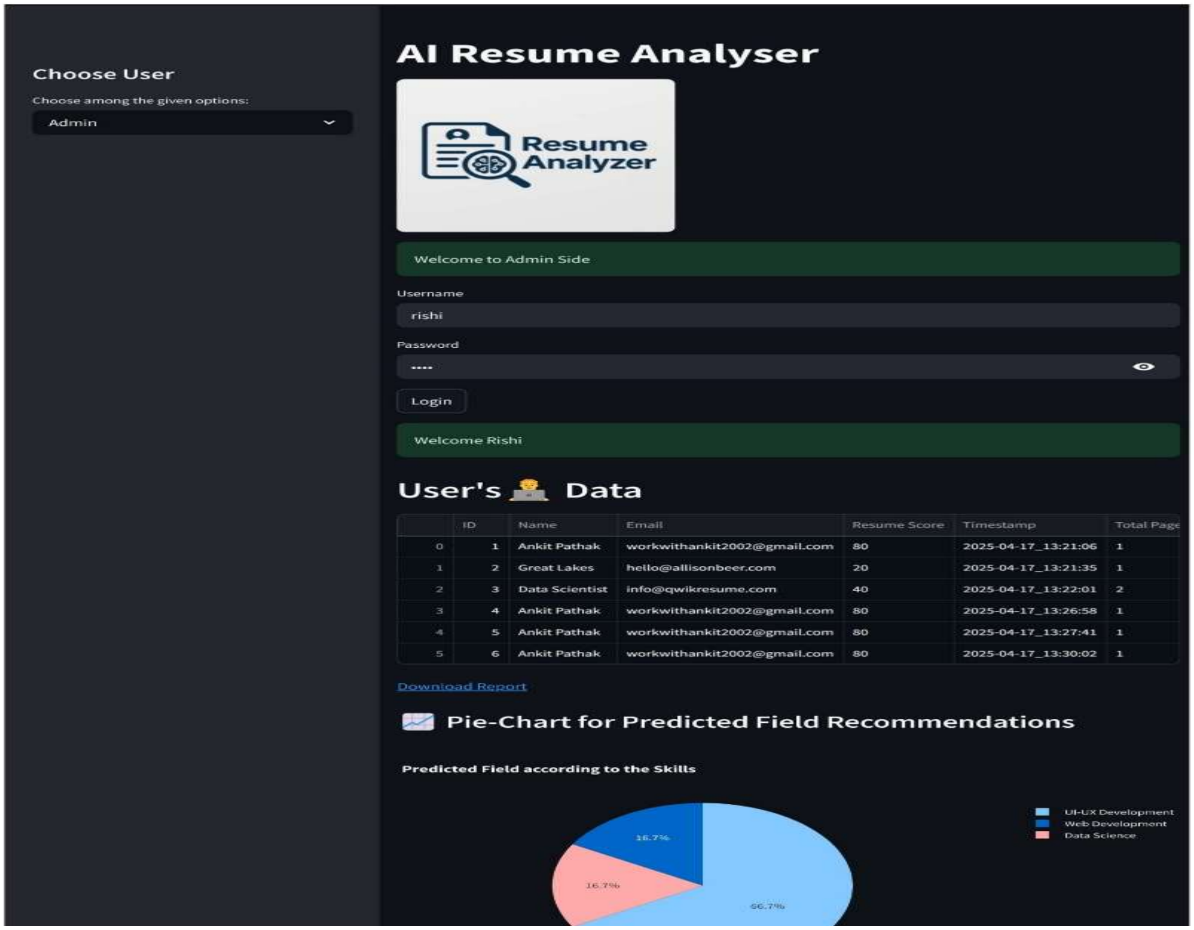
[+] Awesome! You have added your Achievements 🏅

[+] Awesome! You have added your Projects 👨‍💻

Resume Score 📝

Admin Panel – Data Dashboard & Visualization

This segment provides the Admin interface, enabling an authenticated user to view, analyze, and download data collected from normal users. It also includes data visualizations using pie charts to help understand trends across users.



AI Resume Analyser

Welcome to Admin Side

Username: rishi

Password: ····

Welcome Rishi

User's 🧑‍💻 Data

	ID	Name	Email	Resume Score	Timestamp	Total Page
0	1	Ankit Pathak	workwithankit2002@gmail.com	80	2025-04-17_13:21:06	1
1	2	Great Lakes	hello@allisonbeer.com	20	2025-04-17_13:21:35	1
2	3	Data Scientist	info@qwikresume.com	40	2025-04-17_13:22:01	2
3	4	Ankit Pathak	workwithankit2002@gmail.com	80	2025-04-17_13:26:58	1
4	5	Ankit Pathak	workwithankit2002@gmail.com	80	2025-04-17_13:27:41	1
5	6	Ankit Pathak	workwithankit2002@gmail.com	80	2025-04-17_13:30:02	1

📈 Pie-Chart for Predicted Field Recommendations

CONCLUSIONS

The AI Resume Analyzer project successfully demonstrates how artificial intelligence and natural language processing can be integrated into a user-friendly application to automate resume evaluation and provide personalized career guidance.

By leveraging libraries like pyresparser, nltk, and spacy, the system is capable of accurately extracting key information from resumes. It then uses this data to determine a candidate’s experience level, identify relevant skills, and recommend targeted online courses to enhance employability.

The application also features a resume scoring system, which gives users constructive feedback on missing sections such as objectives, projects, and achievements. This helps users improve their resume structure and content, thereby increasing their chances of making a positive impression on recruiters.

The inclusion of a secure Admin Panel allows administrators to monitor user submissions, analyze trends through visualizations (like pie charts), and export data for further insights. This makes the system not just a resume evaluator but a data-driven career advisory tool.

REFERENCES

[1]. AurélienGéron (2017), Hands–On ML with Scikit–Learn and TensorFlow (2nd Edition),Publisher:
O'Reilly Media

[2]. Mirwaes Wahabzada, Anne-Katrin Mahlein, Christian Bauckhage, Ulrike Steiner, Erich Christian Oerke, Kristian Kersting, “'Plant Phenotyping using Probabilistic Topic Models: Uncovering the Hyperspectral Language of Plants,” Scientific Reports 6, Nature, Article number: 22482, 2016.

[3]. Macedo-Cruz A, Pajares G, Santos M, Villegas-Romero I., “Digital image sensor-based assessment of the status of oat (Avena sativa L.) crops after frost damage,” Sensors 11(6), 2011, pp. 6015–6036.

[4]. Yao Q, Guan Z, Zhou Y, Tang J, Hu Y, Yang B, “Application of support vector machine for detecting rice diseases using shape and color texture features,” 2009 International Conference on Engineering Computation, IEEE, Hong Kong, 2009, pp. 79–83.

[5]. Phadikar S, Sil J, “'Rice disease identification using pattern recognition techniques,” IEEE, Khulna, 2008, pp. 420_423.

[6]. Sharada Prasanna Mohanty, David Hughes, Marcel Salathe, “Using Deep Learning for Image- Based Plant Disease Detection,” Computer Vision and Pattern Recognition, to be published.