**Abstract**

In today's competitive job market, students often struggle to identify the right career paths that align with their academic strengths and technical skills. This project aims to predict suitable job profiles for students using machine learning algorithms. The model is trained on a comprehensive dataset containing students' scores in various academic subjects such as Data Structures and Algorithms (DSA), Database Management Systems (DBMS), and Mathematics, along with information about their proficiency in different skills such as Python, JavaScript, and Machine Learning. The dataset comprises both numerical and categorical features, and the target variable is the student's preferred job profile.

Multiple machine learning models, including Decision Trees, Random Forests, Gradient Boosting, and XGBoost, were implemented and evaluated. Through rigorous hyperparameter tuning and performance comparison, the XGBoost model emerged as the most accurate with an overall accuracy of 97%. To facilitate user interaction, a Streamlit-based web application was developed. This app allows users to input their scores and skills through an intuitive interface, after which the trained model predicts their job profile in real-time. The integration of machine learning with a user-friendly web interface ensures that students can access personalized career guidance based on their academic and skill-related data. This project demonstrates the potential of data-driven methods to assist students in making informed career choices.

**Objective**

The primary objective of this project is to develop an intelligent system that predicts a student’s most suitable job profile based on their academic performance and technical skills. With the vast number of career options available in the technology and engineering domains, students often find it challenging to identify roles that best match their strengths and capabilities. This project aims to address this gap by building a machine learning model capable of analyzing students' subject scores and skillsets, thereby providing tailored job profile suggestions.

The system is designed to offer accurate predictions by utilizing both numerical data (e.g., scores in subjects like DSA, DBMS, and Mathematics) and categorical data (e.g., proficiency in skills such as Python, Java, and Machine Learning). Key objectives include:

1. Designing and implementing machine learning models that can classify and predict the student’s most suitable job profile.
2. Optimizing the models through hyperparameter tuning to enhance prediction accuracy.
3. Developing a user-friendly web application using Streamlit that allows students to input their academic and skill data and receive job profile predictions in real-time.
4. Ensuring that the model provides insights based on a combination of both academic and technical skills, making the predictions relevant to the student’s overall competence.

Ultimately, this project seeks to create a tool that empowers students by offering data-driven career guidance, helping them make informed decisions about their future roles in the industry.

**Introduction**

In the ever-evolving world of technology and business, career choices for students have become increasingly diversified. The rapid growth in fields such as Data Science, Software Engineering, Artificial Intelligence, and Cybersecurity has made it crucial for students to align their academic performance and skill development with specific job roles. In particular, engineering students in India face immense pressure to secure placements in a competitive job market, where multiple factors—including academic performance, extracurricular involvement, and technical skills—play a pivotal role in determining their career trajectory.

This project aims to provide an automated system to predict a student’s ideal job profile based on a combination of academic performance in key subjects and proficiency in specific technical skills. By analyzing students' scores in subjects like Data Structures and Algorithms (DSA), Database Management Systems (DBMS), Operating Systems (OS), and Computer Networks (CN), along with their expertise in programming languages and tools such as Python, Java, and Machine Learning, the system can recommend suitable job profiles, such as Software Developer, Data Scientist, or Systems Engineer. This predictive system offers a data-driven approach to assist students in aligning their capabilities with relevant career paths.

**Role of Skills and Academic Subjects in Job Placement**

In the modern job market, recruiters place substantial emphasis on both **academic performance** and **technical skills** when evaluating candidates for various roles. Subjects like DSA, DBMS, OS, and CN are considered core to computer science and engineering curricula, and strong performance in these areas indicates a solid foundation in problem-solving, system design, and software development.

* **Data Structures and Algorithms (DSA):** This subject is crucial for almost all technical roles, particularly software development and system design. It tests a student's ability to optimize solutions, manage resources effectively, and solve complex problems under time constraints. Companies like Google, Amazon, and Microsoft place significant weight on DSA skills during technical interviews.
* **Database Management Systems (DBMS):** DBMS is important for roles such as Database Administrator, Data Engineer, and Backend Developer. It covers the organization, storage, and retrieval of data, which is fundamental to building scalable applications and managing large datasets in industries like banking, e-commerce, and healthcare.
* **Operating Systems (OS):** A strong understanding of OS concepts is crucial for roles like Systems Engineer, Cloud Architect, and DevOps Engineer. This subject covers the management of hardware and software resources, system security, and multitasking, which are vital in environments involving large-scale infrastructures or cloud platforms.
* **Computer Networks (CN):** CN is fundamental for careers in network engineering, cybersecurity, and telecommunications. It equips students with the knowledge to design, implement, and maintain secure communication networks, making it a high-demand skill in today's digital world.

Beyond academic subjects, **technical skills** like proficiency in programming languages and tools—such as Python, Java, and Machine Learning—are critical in shaping career prospects. Skills in **Python** and **Machine Learning** open doors to high-growth areas like Data Science and AI, while expertise in **Java** is still highly valued in enterprise application development. Other key skills such as proficiency in **problem-solving**, **creativity**, and participation in **hackathons** can significantly boost employability by showcasing a student’s ability to innovate and work under pressure.

**Job Profiles in the Technology Sector**

The technology sector offers a wide array of job profiles, each requiring different combinations of academic knowledge and skills. Some of the most sought-after roles include:

* **Software Developer:** This role involves designing, building, and maintaining software applications. It requires strong programming skills, knowledge of algorithms, and the ability to work with multiple technologies and frameworks. Developers can specialize in areas like web development, mobile app development, or enterprise software solutions.
* **Data Scientist/Analyst:** Data Scientists analyze and interpret complex datasets to help businesses make data-driven decisions. This role demands skills in statistics, data mining, and machine learning, along with proficiency in programming languages such as Python and R.
* **Systems Engineer/Administrator:** Systems Engineers work on the design and management of complex systems in both hardware and software environments. This role requires deep knowledge of operating systems, networking, and systems integration.
* **DevOps Engineer:** DevOps Engineers bridge the gap between development and operations, ensuring efficient deployment and integration of systems. They are skilled in automation tools, cloud services, and continuous integration/continuous deployment (CI/CD) pipelines.
* **Cybersecurity Analyst:** Cybersecurity professionals ensure the protection of data and systems from digital attacks. They need expertise in encryption, network security, and vulnerability assessment.

**Placement Trends in Indian Universities**

Placements in Indian engineering colleges, especially for computer science and IT students, have become a key indicator of success. According to placement reports from top institutes like the Indian Institutes of Technology (IITs) and National Institutes of Technology (NITs), the demand for roles in **Software Development**, **Data Science**, and **AI** has surged in recent years. For example, IIT Bombay’s 2023 placement report shows that **50%** of students were recruited into software development roles, while **20%** entered data science positions. The average package for software developers ranged from ₹10 to ₹15 lakhs per annum, with top companies offering even higher packages.

Placement trends also show an increased demand for specialized roles in **AI**, **Cloud Computing**, and **Cybersecurity**, which are among the fastest-growing fields. With the rise of digital transformation initiatives across industries, there is a clear shift towards job profiles that require advanced technical skills in automation, big data, and cloud technologies. The emergence of start-ups and tech unicorns in India has further broadened the range of opportunities available to students, especially those with niche skills like machine learning, data engineering, and full-stack development.

Interestingly, recruiters are not solely focused on technical proficiency. Increasingly, employers are looking for **well-rounded candidates** who demonstrate creativity, leadership, and adaptability. Participation in **extracurricular activities**, such as hackathons, coding competitions, and internships, is often seen as a differentiator in hiring decisions. This aligns with the observation that students with high problem-solving skills and experience in **real-world projects** are more likely to secure top-tier job offers, as they bring practical knowledge to the table.

**Need for Data-Driven Job Profile Prediction**

Given the variability in student performance, skills, and career preferences, there is a need for a systematic and data-driven approach to help students identify the most suitable job profiles. Current methods of job placement guidance often rely on manual career counseling, which can be subjective and limited in scope. By building a machine learning model that considers both academic performance and technical skills, this project aims to automate the process of predicting the most appropriate career paths for students.

Through the analysis of academic scores, participation in technical activities, and skills in demand, this project hopes to contribute towards a more **personalized and accurate career guidance system**. Such a system can benefit students across universities by providing them with data-driven insights into their career possibilities, thereby helping them make more informed decisions about their future.

**Theoretical Background**

The field of machine learning has revolutionized the way data is analyzed, allowing computers to learn from data and make decisions without being explicitly programmed. In this project, we use various machine learning techniques to predict job profiles for students based on their academic performance and skills. To understand the foundation of this project, we need to delve into several key areas, including the types of machine learning, classification models, data preprocessing techniques, and model evaluation methods.

**1. Machine Learning Overview**

Machine learning is a subset of artificial intelligence that involves the development of algorithms that can learn from data and improve their performance over time. There are three primary types of machine learning:

* **Supervised Learning**: In supervised learning, the model is trained on a labeled dataset, meaning that the input features (such as academic scores and skills) are paired with the correct output labels (job profiles). The objective of the model is to learn the mapping from inputs to outputs and make accurate predictions for unseen data. This project is a classic example of supervised learning, as the dataset contains both input features and the corresponding job profiles.
* **Unsupervised Learning**: In unsupervised learning, the model is provided with input data but without any corresponding output labels. The goal is to identify patterns or structures within the data, such as clustering similar data points together. While unsupervised learning is not directly applicable to this project, it is commonly used in areas like data exploration and feature extraction.
* **Reinforcement Learning**: Reinforcement learning involves training models to make decisions in a sequential manner by receiving feedback in the form of rewards or penalties. It is mainly used in tasks such as robotics and game theory, and is not a focus of this project.

In this project, we use **supervised learning** to predict a student’s job profile based on a combination of numerical and categorical features.

**2. Classification in Machine Learning**

The primary task in this project is **classification**, where the goal is to assign a class label (job profile) to a given set of input features (student scores and skills). The classification models are trained on a dataset containing labeled examples of students with known job profiles, and the models learn to predict the job profile for new, unseen students.

Several classification algorithms are used in this project, each with its strengths and limitations:

* **Decision Tree Classifier**: A decision tree is a flowchart-like model where each internal node represents a decision based on the value of a feature, and each leaf node represents a predicted class (job profile). Decision trees are easy to interpret and can handle both numerical and categorical data. However, they are prone to overfitting, especially when the tree becomes too deep.
* **Random Forest Classifier**: A random forest is an ensemble method that combines multiple decision trees to create a more robust and accurate model. Each tree is trained on a random subset of the data and the features, and the final prediction is made by aggregating the predictions of all the trees. Random forests reduce overfitting and improve generalization but can be computationally intensive.
* **Gradient Boosting Classifier (GBM)**: Gradient boosting is another ensemble technique that builds models sequentially, with each new model trying to correct the errors of the previous ones. This approach leads to high accuracy, but it can be slow to train and is sensitive to hyperparameters.
* **XGBoost**: XGBoost, or Extreme Gradient Boosting, is an advanced implementation of the gradient boosting algorithm. It incorporates techniques like regularization to prevent overfitting and parallel processing to speed up training. XGBoost is highly efficient and typically achieves better performance than other ensemble methods. It is the final model chosen for this project due to its accuracy and scalability.

**3. Data Preprocessing Techniques**

Before training any machine learning model, it is essential to preprocess the data to ensure that it is in a suitable format for analysis. Data preprocessing is a crucial step because raw data often contains inconsistencies, missing values, or features that need to be transformed. The preprocessing steps in this project include:

* **Handling Missing Values**: Missing values can occur in any dataset due to incomplete records. In this project, the dataset is first checked for missing values, which, if present, would be handled by either imputing them or removing the incomplete rows, depending on the situation.
* **Feature Scaling**: Feature scaling is the process of normalizing or standardizing numerical features so that they fall within a similar range. In this project, student scores (e.g., in DSA, DBMS, etc.) are scaled using **StandardScaler** to ensure that no single feature dominates the others during model training. Scaling is particularly important when using algorithms like gradient boosting or XGBoost, which are sensitive to feature magnitude.
* **One-Hot Encoding**: The dataset includes categorical features, such as the technical skills (e.g., Python, Java, etc.). These categorical features must be converted into numerical form so that the machine learning models can process them. One-hot encoding is a method used to transform categorical variables into binary vectors. For example, the skill "Python" might be encoded as [0, 1], and "Java" as [1, 0].
* **Label Encoding**: The target variable in this project is the job profile, which is a categorical feature. To train the model, this feature is label encoded, meaning each job profile is assigned a unique numerical value. For example, "Data Scientist" could be encoded as 0, "Software Developer" as 1, and so on.

**4. Model Evaluation Metrics**

Once the models are trained, it is important to evaluate their performance using appropriate metrics. The evaluation metrics used in this project include:

* **Accuracy**: Accuracy is the proportion of correct predictions made by the model. While it is a simple metric, it may not always be the best indicator of performance, especially when the classes are imbalanced.
* **Precision and Recall**: Precision is the proportion of true positive predictions among all positive predictions, while recall is the proportion of true positive predictions among all actual positive instances. Precision and recall are important when dealing with imbalanced datasets, as they provide insights into how well the model distinguishes between different job profiles.
* **F1 Score**: The F1 score is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall, making it useful when there is a trade-off between the two.
* **Confusion Matrix**: A confusion matrix is a table that shows the number of true positive, true negative, false positive, and false negative predictions. It provides a detailed breakdown of the model’s performance across different classes and helps identify where the model is making mistakes.
* **ROC Curve and AUC**: The Receiver Operating Characteristic (ROC) curve plots the true positive rate (recall) against the false positive rate. The Area Under the Curve (AUC) is a measure of the model’s ability to distinguish between different classes. A higher AUC indicates a better-performing model.

**5. Streamlit for Deployment**

The deployment of the machine learning model in this project is done using **Streamlit**, an open-source framework for creating interactive web applications for machine learning and data science projects. Streamlit simplifies the process of building a user interface and allows users to interact with the model by inputting their academic scores and technical skills. The following components of the Streamlit application are essential:

* **User Input Forms**: The application includes input fields such as sliders and dropdowns, where users can enter their academic scores and select their technical skills.
* **Prediction Output**: Once the user inputs their data, the machine learning model predicts the job profile, which is displayed on the web interface in real-time.
* **Model Integration**: The trained machine learning model (XGBoost) is integrated into the Streamlit application using the joblib library, allowing the app to make predictions based on the user’s input.

The theoretical foundation of this project lies in the concepts of supervised learning, specifically classification, as well as data preprocessing techniques like feature scaling and encoding. The models chosen—Decision Trees, Random Forests, Gradient Boosting, and XGBoost—are powerful classification tools that have been fine-tuned to achieve accurate job profile predictions. The use of Streamlit for deployment further enhances the project by providing a user-friendly interface for real-time interaction with the model.

By combining these advanced machine learning techniques with effective data preprocessing and model evaluation, this project aims to offer a reliable and data-driven solution for predicting job profiles based on student performance and skillsets.

**Hardware & Software Requirements**

In order to successfully implement the "Job Profile Prediction Based on Skills and Academic Performance" project, the system requires both hardware and software configurations that support machine learning model development, data processing, and web application deployment. Below are the detailed requirements:

**Hardware Requirements**

To ensure efficient execution of the machine learning algorithms and a smooth experience with the data-intensive tasks, the following hardware configuration is recommended:

1. **Processor:**
   * **Intel Core i5 (or higher)** or **AMD Ryzen 5 (or higher)**.
   * A multi-core processor with a clock speed of 2.5 GHz or above is essential for faster data processing and model training.
2. **RAM:**
   * **4 GB RAM (minimum)**, with **8 GB RAM (recommended)**.
   * Machine learning operations, such as training models on large datasets, and running web applications like Streamlit, require significant memory resources to avoid performance bottlenecks.
3. **Storage:**
   * **256 GB SSD (recommended)** or **1 TB HDD**.
   * Solid-state drives (SSD) are preferred for faster read/write speeds, especially during data processing and saving/loading machine learning models.
4. **Graphics Processing Unit (GPU) (Optional):**
   * For handling large datasets and more complex machine learning models like deep learning, a **dedicated GPU** such as **NVIDIA GTX 1060 or higher** may be beneficial. However, for this specific project, the GPU is optional since the models used (Random Forest, Decision Tree, etc.) do not necessarily require GPU acceleration.
5. **Operating System:**
   * **Windows 10/11**, **macOS** (version 10.13 or later), or **Linux (Ubuntu 20.04 LTS or later)**.

**Software Requirements**

The software components required for this project include tools for data preprocessing, machine learning model development, and deployment. Here’s a breakdown of the necessary software:

1. **Programming Language:**
   * **Python 3.7 or later**.
   * Python is the core programming language used for data manipulation, machine learning model development, and web application deployment in this project. Libraries like pandas, scikit-learn, XGBoost, and Streamlit are utilized for various tasks.
2. **Libraries and Frameworks:**
   * **Pandas:**
     + For data manipulation, cleaning, and analysis of the student dataset.
   * **NumPy:**
     + Used for numerical computations, handling arrays, and performing operations on datasets.
   * **Scikit-learn:**
     + Provides implementations of machine learning algorithms such as Decision Trees, Random Forests, and data preprocessing techniques like scaling and encoding.
   * **XGBoost:**
     + For implementing the XGBoost algorithm, which is known for high accuracy and efficiency in classification tasks.
   * **Matplotlib/Seaborn:**
     + Used for data visualization, such as plotting distribution graphs, confusion matrices, and learning curves.
   * **Joblib:**
     + To save and load trained machine learning models, making it easier to use the model in production.
   * **Streamlit:**
     + An open-source framework used to develop the web interface for user interaction. It allows users to input their academic scores and skills and receive job profile predictions in real-time.
3. **Integrated Development Environment (IDE):**
   * **Jupyter Notebook:**
     + For developing and experimenting with the machine learning models, performing data preprocessing, and visualizing results.
   * **VS Code / PyCharm / Spyder:**
     + For writing and debugging Python code, especially during the development of the app.py for the Streamlit web application.
4. **Version Control:**
   * **Git / GitHub:**
     + For tracking code changes and version control. GitHub is also used for collaboration and code repository hosting, allowing easier deployment and sharing of code with other contributors.
5. **Database Management (Optional):**
   * **SQLite / MySQL:**
     + If storing user data or making the project more dynamic, a database such as SQLite or MySQL can be used for storing past predictions or user information.
6. **Model Deployment:**
   * **Heroku / AWS / GCP (Optional):**
     + For deploying the Streamlit web application to a cloud platform. Heroku offers an easy-to-use platform for hosting Python web apps, while AWS and GCP provide scalable infrastructure for larger applications.
7. **Other Dependencies:**
   * **LabelEncoder/OneHotEncoder:**
     + Used to convert categorical data (such as skills) into numerical form for model training.
   * **StandardScaler:**
     + Required for normalizing numerical data to ensure that all features contribute equally during model training.

**Optional Hardware/Software Add-ons:**

1. **GPU-Accelerated Development (Optional):**
   * If you are working with larger datasets or plan to extend the project with more computationally demanding models like deep learning, using a GPU for accelerated computations (e.g., via **CUDA** or **TensorFlow** on NVIDIA GPUs) can significantly reduce training time.
2. **Anaconda Distribution (Optional):**
   * Anaconda simplifies the management of Python packages and environments, especially when handling various dependencies for machine learning and data science projects.
3. **Docker (Optional):**
   * If deploying the project in a production environment, Docker can be used to containerize the application, ensuring consistent behavior across different environments by packaging the app along with its dependencies.

**Summary of Requirements:**

| **Category** | **Specification/Tool** |
| --- | --- |
| **Processor** | Intel i5 or Ryzen 5 (or higher) |
| **RAM** | 8 GB (min) / 16 GB (recommended) |
| **Storage** | 512 GB SSD (recommended) / 1 TB HDD |
| **Operating System** | Windows 10/11, macOS, Linux (Ubuntu 20.04 or later) |
| **Python** | Python 3.7 or later |
| **IDE** | Jupyter, VS Code, PyCharm |
| **Libraries** | Pandas, NumPy, Scikit-learn, XGBoost, Matplotlib |
| **Web Framework** | Streamlit |
| **Version Control** | Git, GitHub |
| **Optional Software** | Docker, SQLite/MySQL, Heroku, AWS, GCP |

**METHODOLOGY**

**1. Importing Required Libraries**

python

Copy code

import warnings

warnings.filterwarnings('ignore')

* **Purpose:** Suppresses any warning messages that could clutter the output. Warnings are often related to deprecated features or non-critical issues that don't affect code execution.

python

Copy code

import os

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

* **Purpose:** Imports essential libraries for data handling and visualization:
  + os: For interacting with the operating system (e.g., file paths).
  + numpy: Handles numerical operations, especially arrays and matrices.
  + pandas: Manages and manipulates structured data in DataFrames.
  + matplotlib and seaborn: For data visualization (e.g., plots, histograms, and heatmaps).

**2. Machine Learning Libraries**

python

Copy code

from sklearn.model\_selection import train\_test\_split, GridSearchCV, learning\_curve

from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler, label\_binarize

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, roc\_curve, auc

from xgboost import XGBClassifier # XGBoost

from sklearn.multiclass import OneVsRestClassifier

* **Purpose:** Imports machine learning functions and metrics from scikit-learn and xgboost:
  + train\_test\_split: For splitting the dataset into training and test sets.
  + GridSearchCV: Used to perform hyperparameter tuning for ML models.
  + learning\_curve: Helps visualize the learning progress of a model.
  + LabelEncoder, OneHotEncoder, StandardScaler: Preprocessing tools for encoding categorical data and scaling numerical features.
  + DecisionTreeClassifier, RandomForestClassifier, GradientBoostingClassifier, XGBClassifier: Classifiers used to predict the target variable.
  + accuracy\_score, confusion\_matrix, etc.: Metrics for evaluating model performance.
  + OneVsRestClassifier: For handling multi-class classification using a strategy that treats the classification as multiple binary problems.

python

Copy code

import joblib

* **Purpose:** joblib is used to save and load trained models for future use without retraining them.

**3. Loading the Dataset**

python

Copy code

data = pd.read\_csv('../Dataset/StudentPlacement.csv')

data.tail()

* **Purpose:** Loads the student placement dataset from a CSV file using pandas. The tail() function shows the last five rows of the dataset for a quick look at its structure.

**4. Understanding the Dataset**

python

Copy code

# shape of the dataset

data.shape

* **Purpose:** Displays the number of rows and columns in the dataset.

python

Copy code

# basic info of the dataset

data.info()

* **Purpose:** Provides details like data types of columns, non-null values, and memory usage of the dataset.

python

Copy code

# checking for any null values

data.isnull().sum()

* **Purpose:** Checks each column for missing (null) values.

python

Copy code

# checking for any duplicate values in the dataset

data[data.duplicated(data.columns[:-1])]

* **Purpose:** Identifies any duplicate records in the dataset, excluding the target column ('Profile').

**5. Exploratory Data Analysis (EDA)**

python

Copy code

# viewing the unique profiles that this dataset have

data['Profile'].unique()

* **Purpose:** Lists all unique values in the target column ('Profile'), which represents different job profiles.

python

Copy code

# Extracting the numerical and categorical features

numerical\_data = data.select\_dtypes(include=['number'])

categorical\_data = data.select\_dtypes(include=['object', 'category'])

* **Purpose:** Separates numerical and categorical columns. numerical\_data includes features like scores, while categorical\_data includes features like 'Skill 1' and 'Skill 2'.

python

Copy code

# no of unique values does each numberical feature haves

numerical\_data.nunique()

* **Purpose:** Shows the number of unique values in each numerical column.

**6. Correlation Heatmap**

python

Copy code

sns.set(rc={'figure.figsize':(10, 6)})

sns.heatmap(numerical\_data.corr(), annot = True)

plt.show()

* **Purpose:** Creates a correlation heatmap to visualize the relationships between numerical features (e.g., DSA, DBMS, etc.).

**7. Distribution Plots**

python

Copy code

plt.figure(figsize=(16, 12))

for i, col in enumerate(numerical\_cols, 1):

plt.subplot(5, 2, i)

sns.histplot(data[col], kde=True, bins=20)

plt.title(f'Distribution of {col}')

plt.xlabel(col)

plt.ylabel('Frequency')

plt.tight\_layout()

plt.show()

* **Purpose:** Plots histograms for each numerical column to analyze the distribution of values.

**8. Box Plots for Outlier Detection**

python

Copy code

plt.figure(figsize=(7, 20))

for i, col in enumerate(numerical\_cols, 1):

plt.subplot(5, 2, i)

sns.boxplot(y=data[col])

plt.title(f'Box Plot of {col}')

plt.ylabel(col)

plt.tight\_layout()

plt.show()

* **Purpose:** Plots box plots for each numerical column to identify outliers in the dataset.

**9. Outlier Removal Using Interquartile Range (IQR)**

python

Copy code

def remove\_outliers(df, columns):

for col in columns:

Q1 = df[col].quantile(0.25)

Q3 = df[col].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

df = df[(df[col] >= lower\_bound) & (df[col] <= upper\_bound)]

return df

cleaned\_data = remove\_outliers(data.copy(), numerical\_cols)

* **Purpose:** Removes outliers from the dataset by calculating the interquartile range (IQR) and filtering values that fall outside a reasonable range (1.5 \* IQR).

**10. Pie Chart of Profile Distribution**

python

Copy code

plt.figure(figsize=(7, 20))

cleaned\_data['Profile'].value\_counts().plot.pie()

plt.show()

* **Purpose:** Plots a pie chart to show the distribution of the target variable (Profile).

**11. One-Hot Encoding for Categorical Features**

python

Copy code

skill\_encoder = OneHotEncoder(handle\_unknown='ignore')

skills\_encoded = skill\_encoder.fit\_transform(cleaned\_data[['Skill 1', 'Skill 2']]).toarray() \* 2.0

skills\_encoded\_df = pd.DataFrame(skills\_encoded, columns=skill\_encoder.get\_feature\_names\_out(['Skill 1', 'Skill 2']))

* **Purpose:** Encodes the categorical columns ('Skill 1' and 'Skill 2') using one-hot encoding and gives them more weight (multiplies by 2.0). Creates a DataFrame skills\_encoded\_df to store the encoded features.

**12. Combining Numerical and Encoded Data**

python

Copy code

numerical\_features = cleaned\_data.drop(columns=['Skill 1', 'Skill 2', 'Profile'])

X = pd.concat([numerical\_features.reset\_index(drop=True), skills\_encoded\_df.reset\_index(drop=True)], axis=1)

* **Purpose:** Combines the numerical features and the one-hot encoded categorical features into a single feature matrix X.

**13. Feature Scaling**

python

Copy code

scaler = StandardScaler()

X[X.columns[:10]] = scaler.fit\_transform(X[X.columns[:10]])

* **Purpose:** Scales the first 10 numerical columns to ensure they are standardized (mean of 0 and standard deviation of 1), which is important for distance-based algorithms.

**14. Encoding the Target Variable**

python

Copy code

profile\_encoder = LabelEncoder()

cleaned\_data['Profile'] = profile\_encoder.fit\_transform(cleaned\_data['Profile'])

* **Purpose:** Converts the categorical target column ('Profile') into numerical labels using LabelEncoder.

**15. Splitting the Dataset**

python

Copy code

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=42, shuffle=True, test\_size=0.3, stratify=y)

* **Purpose:** Splits the dataset into training (70%) and test (30%) sets. The stratify parameter ensures that the proportion of different profiles in the training and test sets is the same.