PROJECT REPORT

Performance Comparison of Machine Learning Algorithms for Efficient Resume Job Matching

Submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by

NAME: Siddharth Singh Verma **University Roll No- 2119248**

NAME: Atul Kumar **University Roll No- 2118370**

NAME: Atul Chaudhary **University Roll No- 2118368**

Under the guidance of

Mr. Mukesh Kumar

Assistant Professor

Project Group No: 155

Department of Computer Science and Engineering

Graphic Era Hill University

June, 2025

Graphic Era HILL UNIVERSITY
Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)
University under section 2(f) of UGC Act, 1956

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project progress report entitled **"Performance Comparison of Machine Learning Algorithms for Efficient Resume Job Matching"** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering in the Department of Computer Science and Engineering of the Graphic Era Hill University, Dehradun shall be carried out by the undersigned under the supervision of **Mr. Mukesh Kumar, Assistant Professor**, Department of Computer Science and Engineering, Graphic Era Hill University, Dehradun.

Name: Siddharth Singh Verma	University Roll no.: 2119248	signatures
Name: Atul Kumar	University Roll no.: 2118370	signatures
Name: Atul Chaudhary	University Roll no.: 2118368	signatures

The above-mentioned students shall be working under the supervision of the undersigned on the **"Performance Comparison of Machine Learning Algorithms for Efficient Resume Job Matching"**

Guide: **Head of the Department**

Date:

Place:

ACKNOWLEDGEMENT

We would like to express our gratitude to the Almighty, the most beneficent and the most merciful, for successful completion of the project.
We would like to express our deepest appreciation to all those who provided us the possibility to complete this project.
A special gratitude we give to our project guide, Mr. Mukesh Kumar, whose contribution in stimulating suggestions and encouragement, helped us to coordinate our project especially in writing this report.
We wish to thank our parents for their continuing support and encouragement.
We also wish to thank them for providing us with the opportunity to reach this far in our studies.
At last, but not the least, we are greatly indebted to others who directly or indirectly helped us during this project.

Name of student 1: Siddharth Singh Verma
University Roll no.: 2119248

Name of student 2: Atul Kumar
University Roll no. 2118370

Name of student 2: Atul Chaudhary
University Roll no. 2118368

ABSTRACT

Effective resume-job matching is crucial in modern recruitment processes, helping organizations identify candidates whose skills and qualifications align with job requirements. This project systematically explores various approaches to automate and improve this matching process. Beginning with rule-based techniques that rely on exact keyword and skill matching, we gradually transition to more sophisticated methods, including machine learning (ML), deep learning, and generative AI. By evaluating the performance of these approaches, we aim to determine the most efficient and accurate method for enhancing automated hiring systems. The study not only assesses traditional ML models such as Logistic Regression, Random Forest, and XGBoost but also integrates advanced natural language processing (NLP) models like BERT and GPT. Ultimately, we will try to explore the potential of generative AI in refining job-resume compatibility, improving dataset augmentation, and optimizing recruitment efficiency

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

Abbreviation	Full Form
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
NLP	Natural Language Processing
ANN	Artificial Neural Network
ATS	Applicant Tracking System
LLM	Large Language Model
TF-IDF	Term Frequency-Inverse Document Frequency

CHAPTER 1

INTRODUCTION

Effective resume-job matching is crucial in modern recruitment processes, helping organizations identify candidates whose skills and qualifications align with job requirements. This project focuses on developing and refining techniques for aligning job descriptions with resumes by systematically progressing through various methodologies. Initially, it employs rule-based approaches that rely on predefined criteria to match relevant skills, experience, and qualifications. As the project advances, it incorporates machine learning (ML) models to enhance the accuracy of matching by identifying patterns and relationships in textual data. Ultimately, the project will explore generative AI models, which can generate and refine job-resume matches by leveraging advanced language models. The goal is to evaluate the effectiveness of each approach, comparing their efficiency and accuracy in real-world hiring scenarios.

The project systematically explores various approaches to automate and improve this matching process. Beginning with rule-based techniques that rely on exact keyword and skill matching, we gradually transition to more sophisticated methods, including machine learning (ML), deep learning, and generative AI . By evaluating the performance of these approaches, we aim to determine the most efficient and accurate method for enhancing automated hiring systems. The study not only assesses traditional ML models such as Logistic Regression, Random Forest, and XGBoost but also integrates advanced natural language processing (NLP) models like BERT and GPT. Ultimately, we will try to explore the potential of generative AI in refining job-resume compatibility, improving dataset augmentation, and optimizing recruitment efficiency.

Another related project also evaluated job title prediction from resumes using different approaches. This evaluation started with a rule-based method and compared it to traditional ML models before moving to Transformer-based models(e.g., DistilBERT). The objective was to report accuracy and model performance and visualize and compare all approaches. The assumptions made in this project were that rule-based accuracy would be low (around 20%) due to limited context understanding, traditional ML models using TF-IDF and classifiers like Random Forest, Logistic Regression, SVM, and XGBoost would have an average accuracy of around 80% based on previous experiments, and Transformer models like DistilBERT fine-tuned on resume-to-job-title classification would achieve a significantly higher accuracy (observed ~95.85%).

CHAPTER 2

OBJECTIVE

The primary objective of this project is to develop a robust and intelligent system capable of aligning job descriptions with resumes, thereby improving the recruitment process for both employers and job seekers. As organizations receive an overwhelming number of applications, there is a growing need for automated tools that can efficiently screen resumes and match them with the most relevant job openings. To address this challenge, the project aims to explore and evaluate multiple approaches, ranging from traditional rule-based methods to modern generative AI techniques.

1. Establish a Rule-Based Baseline
The first step in this project involves implementing a rule-based approach that relies on predefined rules and criteria to match resumes with job descriptions. This method is designed to identify the presence of specific keywords, required skills, educational qualifications, and experience levels. By using techniques such as text pattern matching and keyword frequency analysis, the rule-based system provides an interpretable and deterministic starting point for the job-resume alignment task.

2. Integration of Machine Learning Models
While rule-based systems are straightforward, they often lack the ability to understand contextual information or adapt to variations in data. To address this limitation, the project incorporates traditional machine learning (ML) algorithms to improve the accuracy of the matching process. These models learn from labelled examples and identify complex relationships between job descriptions and resumes. By experimenting with classifiers such as Logistic Regression, Random Forest, and XGBoost, the system aims to learn generalizable patterns and improve the performance of the matching process across diverse datasets.

3. Adoption of Advanced NLP Models
To further enhance the semantic understanding of both job descriptions and resumes, the project explores the use of advanced Natural Language Processing (NLP) models, particularly those based on transformer architectures. Contextual embeddings from models such as BERT and sentence transformers are used to capture the nuanced meaning of terms and phrases beyond simple keyword matching. These models help improve the system's ability to recognize relevant matches even when different terminology is used.

4. Exploration of Generative AI Techniques
In addition to traditional models, the project investigates the application of generative AI models, such as GPT or other large language models (LLMs), to perform tasks like resume summarization, job-resume compatibility scoring, and data augmentation. These models can generate human-like text and offer creative ways to align or refine job-resume pairs. Generative AI is also explored for its potential in generating synthetic

datasets, which can be used to train and improve ML models when annotated data is scarce.

5. Comparative Performance Evaluation

A key objective of this project is to evaluate and compare the performance of all the approaches mentioned above. This involves assessing each model or technique based on various metrics such as accuracy, precision, recall, F1-score, computational efficiency, and scalability. By conducting a thorough comparative analysis, the project seeks to identify the trade-offs and ideal use cases for each method.

6. Investigation of Job Title Prediction from Resumes

An associated objective of this project is to develop a system for predicting job titles based on resume content. This is crucial for building intelligent classification and recommendation systems. The approach follows a similar pipeline, starting with rule-based classification, moving through traditional ML models, and culminating in transformer-based models such as DistilBERT. The goal is to evaluate which approach is best suited for this classification task in terms of both accuracy and interpretability.

7. Visualization and Interpretation

To make the results accessible and understandable, the project also aims to incorporate visualization techniques. This includes graphical representations such as performance charts, confusion matrices, and bar graphs that compare the outputs of different models. These visual tools help interpret results and present findings effectively.

CHAPTER 3

PROJECT METHODOLOGY / DESIGN

Our approach for efficient resume job matching follows a structured progression, starting with simple rule-based techniques and advancing toward sophisticated AI-driven models. The key phases are outlined below :

- **Rule-Based Matching**
  - **Overview**: Implementing a simple rule-based matching system to compare resumes and job descriptions to determine the relevance of keywords.
  - **Steps**:
    - Extract key terms from both resumes and job descriptions.
    - Count the number of overlapping words to assess similarity.
    - Normalize scores to compare different resumes fairly.
  - **Advantages**: Easy to implement and interpret; no need for training data.
  - **Limitations**: Ignores word context and synonyms; can be affected by keyword stuffing.
- **Machine Learning (ML) Algorithms**
  - To enhance accuracy beyond basic keyword matching, we train multiple machine learning models to identify deeper relationships between resumes and job descriptions.
  - **Models Used**:
    - **Logistic Regression**: A linear model that predicts the probability of a job match based on weighted features.
    - **Random Forest Classifier**: An ensemble learning method that improves prediction accuracy by combining multiple decision trees.
    - **Support Vector Machine (SVM)**: A model that classifies resumes based on a hyperplane that maximizes the margin between different job categories.
    - **Decision Tree Classifier**: A tree-based approach that splits data based on important features to determine job relevance.
    - **Gradient Boosting Classifier**: A sequential ensemble learning technique that corrects errors in previous iterations to improve prediction accuracy.
    - **XGBoost Classifier**: A highly efficient gradient boosting implementation designed for better performance and faster execution.
  - The effectiveness of each model will be evaluated using relevant metrics such as accuracy and classification reports.
- **Generative AI Enhancements**
  - The final phase involves leveraging large language models (LLMs) to enhance dataset quality and refine job-resume matching.
  - **Planned Activities**:
    - Improve dataset augmentation by generating realistic variations of resumes and job descriptions for training purposes.
    - Enhance job-resume compatibility by utilizing LLMs to suggest missing skills or optimize resume content to better align with job postings.

For the related project on job title prediction:

1. Rule-Based System (Baseline)

A simple rule-based approach was implemented as the baseline. This system matched keywords from resumes to job titles using manually defined heuristics. While straightforward and interpretable, this method faced several limitations:

- Inability to handle **synonyms** (e.g., "developer" vs. "programmer").
- Poor recognition of **role variations** and **contextual meanings**.
- Lack of flexibility when dealing with **noisy or non-standard resume formats**.

2. Traditional Machine Learning Models

To improve upon the limitations of the rule-based system, several traditional machine learning models were evaluated. These models used **TF-IDF** vectorization to convert textual data into numerical features and were trained using:

- **Random Forest**
- **Logistic Regression**
- **Support Vector Machine (SVM)**
- **Decision Tree**
- **XGBoost**

3. Transformer-Based Model: DistilBERT

To incorporate contextual understanding of resume content, the transformer model **DistilBERT** was fine-tuned on a job title classification task. Key points include:

- Model: `distilbert-base-uncased`
- Framework: **HuggingFace Transformers**
- Task: Resume → Job Title classification
- Advantage: Captures semantic relationships and context better than traditional methods.

4. Data Preprocessing

To ensure optimal model performance and standardized input across models, several data preprocessing steps were applied:

- **Text Cleaning**: Removal of irrelevant characters, special symbols, and excessive whitespace.
- **Stopword Removal**: Elimination of common English stopwords (e.g., "the", "is", "and") using NLP libraries.
- **Lemmatization**: Normalization of words to their root forms using **NLTK** (Natural Language Toolkit).
- **TF-IDF Vectorization**: Conversion of resume text into numerical features that represent the importance of words across the dataset.
- **Label Encoding**: Transformation of job titles into numerical labels for compatibility with ML classifiers.

5. Dataset

Two main datasets were used in the project:

- **Updated Resume Dataset**: This dataset contains raw resume texts along with their corresponding job titles. It was used primarily for the classification task.
- **Merged Dataset**: Created by combining the resume dataset with job descriptions. This was used to experiment with rule-based job-resume alignment and explore semantic matching techniques.

CHAPTER 4

RESULT AND IMPLEMENTATION

Results of Rule-Based Approach

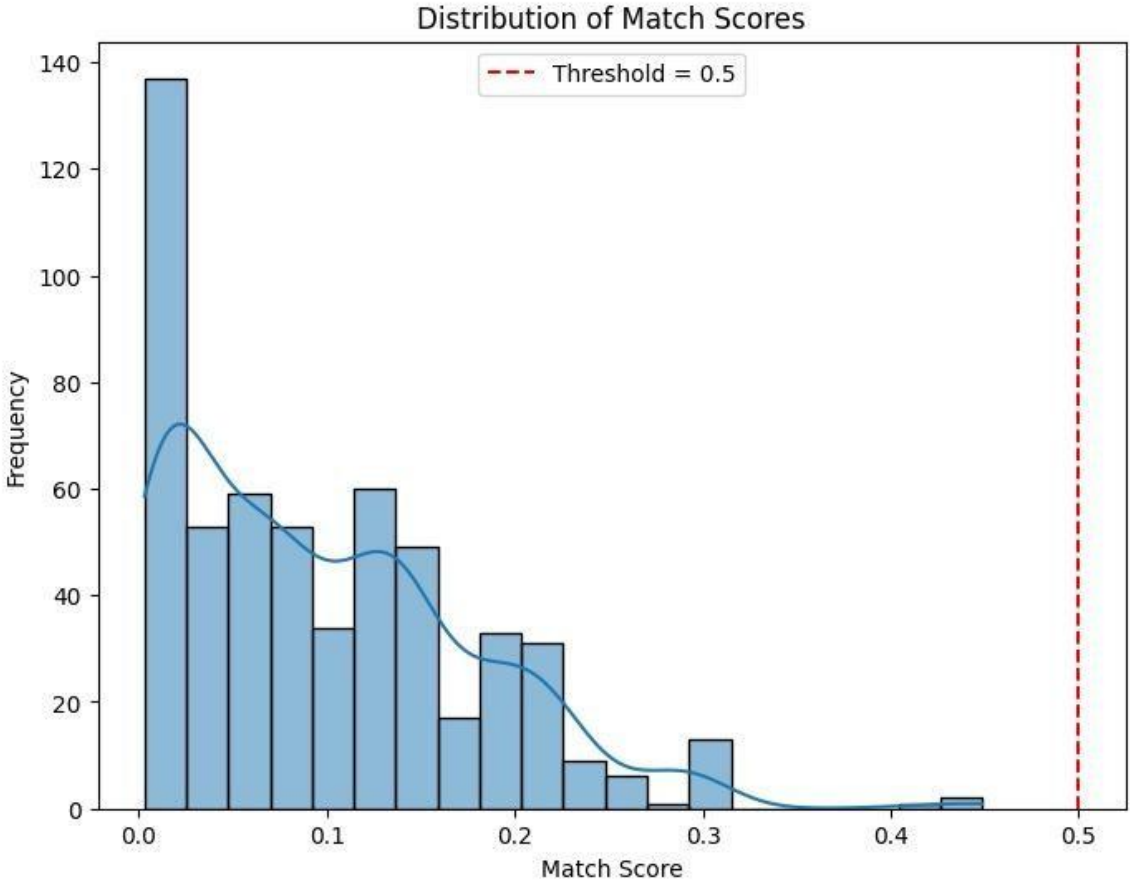

Figure - 4.1

• **Most Scores Are Low**: The majority of match scores are close to 0.0, indicating minimal keyword overlap between resumes and job descriptions.
• **Few High Scores**: Only a small number of resume-job pairs have a high match score, suggesting that keyword matching alone may not capture strong similarities.
• **Threshold Consideration**: The red dashed line at 0.5 represents a potential threshold for a good match, but very few resumes reach this level.
• **Skewed Distribution**: The distribution is right-skewed, meaning most matches are weak, with only a few strong matches.
• **Implications**: Keyword matching alone is likely insufficient for effective resume-job matching. Lowering the threshold might increase the number of matches, but likely with lower accuracy.

Result of ML Models

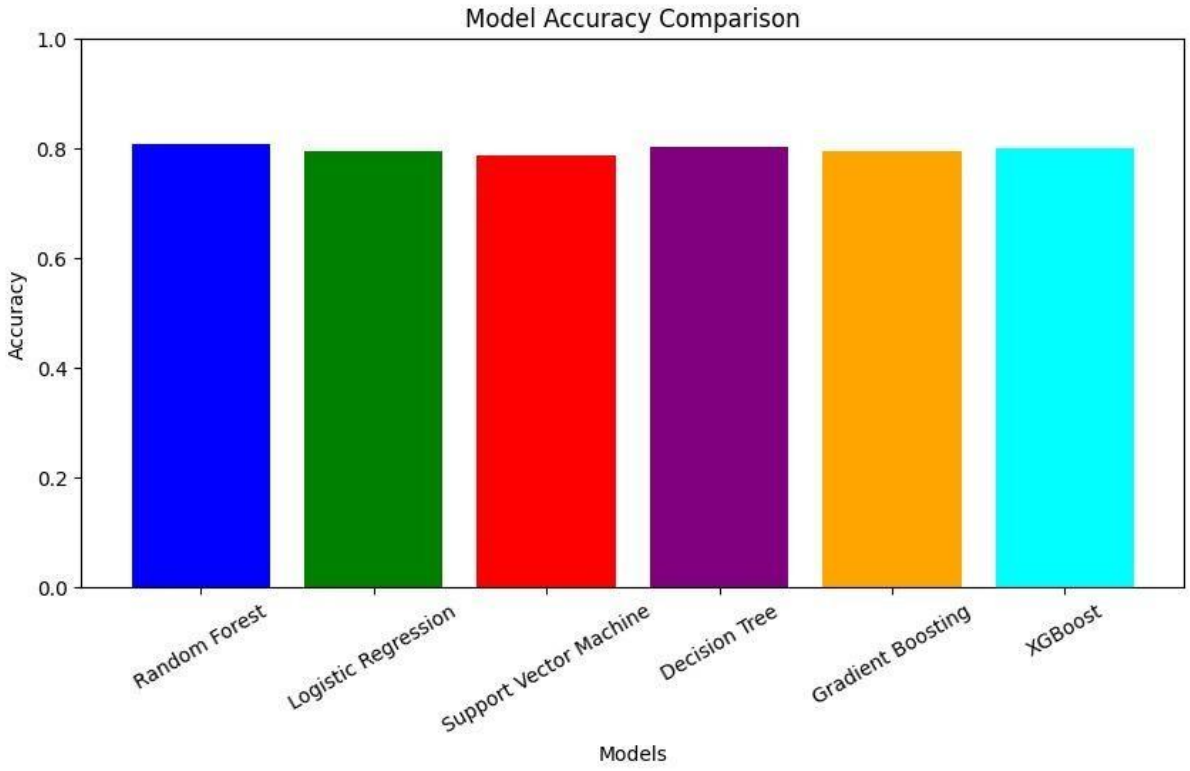

Figure - 4.2

• **Similar Performance Across Models**: All models achieve an accuracy of approximately 78% to 82%. No model significantly outperforms the others.
• **Random Forest & XGBoost Perform Slightly Better**: Random Forest and XGBoost show slightly higher accuracy compared to other models. This is expected, as ensemble methods like these tend to generalize better and reduce overfitting.
• **Support Vector Machine (SVM) & Logistic Regression**: These models perform slightly lower than Random Forest and XGBoost. Since SVM and Logistic Regression are linear models, they may struggle with capturing complex patterns in the dataset.
• **Decision Tree & Gradient Boosting**: Their performance is comparable to other models. Decision Tree might have overfitted if pruning was not applied. Gradient Boosting is expected to perform well, but fine-tuning hyperparameters may further improve results.
ML Model Accuracy: All traditional ML models (Random Forest, Logistic Regression, SVM, Decision Tree, XGBoost) clustered around 78%–81% accuracy, with an average of ~80%.

Result of Transformer Model:

Transformer Model (DistilBERT):
Achieved a final validation accuracy of 95.85% after fine-tuning9 . Training loss decreased consistently, and validation loss also dropped significantly (final: 0.95).

Epoch	Training Loss	Validation Loss
1	3.1371	2.63923
2	2.4617	1.931271
3	1.7788	1.37421
4	1.2819	1.053812
5	1.0394	0.953171

Table – 4.1

Visual Analysis:

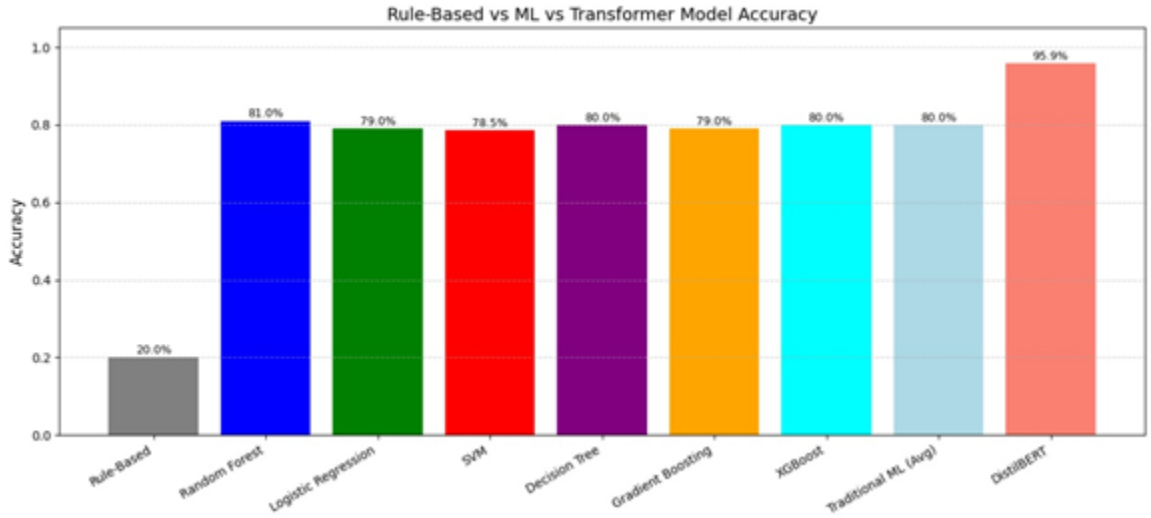

Figure – 4.3

A combined bar chart clearly showed performance improvement from rule-based to ML to Transformer models9 .
Implementation details are not explicitly detailed in the provided excerpts beyond the methodologies and preprocessing steps.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

The results indicate that **keyword matching alone (rule-based approach) is likely insufficient for effective resume-job matching**, as demonstrated by the low match scores observed. While traditional **machine learning models achieve a decent level of accuracy (around 78%-82%)**, they might be limited in capturing complex patterns in the data. Ensemble methods like **Random Forest and XGBoost showed slightly better performance** compared to linear models.

The related project on job title prediction further highlights these trends, with a **low accuracy for the rule-based system (20%)** and an average of **80% accuracy for traditional ML models**. Notably, **Transformer-based models like DistilBERT significantly outperformed both rule-based and traditional ML methods**, achieving a high accuracy of 95.85% in job title prediction due to better context understanding. This suggests that leveraging advanced natural language processing techniques is crucial for improving the accuracy of automated hiring systems.

Future Scope:

• Add job description → resume matching in a similar pipeline, likely leveraging Transformer models given their superior performance in job title prediction.

• Use LLMs like Gemini/GPT for additional features such as resume improvement suggestions and custom job fit feedback. This aligns with the initial project's plan to explore generative AI for enhancing job-resume compatibility and dataset augmentation.

• Build a streamlined UI in Django or Colab (Gradio/Streamlit) for demo/testing to make the system more accessible.

The initial project also intends to further explore generative AI for improving dataset augmentation and optimizing recruitment efficiency. Comparing the performance of various machine learning and generative AI models on the job-resume matching task will be a key aspect of future work.

APPENDIX A

(Code)

Code of Rule Base Approach:

```python
import pandas as pd
from collections import Counter

# Function to perform keyword matching
def keyword_matching(resume, job_description):
    if pd.isna(resume) or pd.isna(job_description):
        return 0  # No match if any is empty

    resume_words = set(resume.lower().split())
    job_desc_words = set(job_description.lower().split())

    common_words = resume_words.intersection(job_desc_words)

    return len(common_words) / max(len(job_desc_words), 1)  # Normalize by job description length

# Load dataset
new_df = pd.read_csv("/content/drive/MyDrive/finalyearproject/merged_dataset.csv")  # Update with actual path

# Apply keyword matching
new_df["Match Score"] = new_df.apply(lambda row: keyword_matching(row["Resume"], row["Job Description"]), axis=1)

# Display results
print(new_df.head())
```

Code of Machine Leaning Algorithms:

1. Logistic Regression:

```python
# Logistic Regression model with class balancing
log_reg = LogisticRegression(max_iter=1000, class_weight="balanced")

# Hyperparameter tuning using GridSearchCV
param_grid = {"C": [0.01, 0.1, 1, 10], "max_iter": [500, 1000, 1500]}
grid_search = GridSearchCV(log_reg, param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)

# Get the best Logistic Regression model
best_log_reg = grid_search.best_estimator_

# Predictions
y_pred_log_reg = best_log_reg.predict(X_test)

# Evaluation
log_reg_accuracy = accuracy_score(y_test, y_pred_log_reg)
print("Logistic Regression Accuracy:", log_reg_accuracy)
print("Logistic Regression Classification Report:\n", classification_report(y_test, y_pred_log_reg))
```

2. RandomForrest:

```
#Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, class_weight="balanced", random_state=42)
rf_model.fit(X_train, y_train)

# Predictions for Random Forest
y_pred_rf = rf_model.predict(X_test)

# Evaluation for Random Forest
rf_accuracy = accuracy_score(y_test, y_pred_rf)
print("Random Forest Accuracy:", rf_accuracy)
print("Random Forest Classification Report:\n", classification_report(y_test, y_pred_rf))
```

3. Support Vector Machine

```
[ ] from sklearn.svm import SVC

    # Support Vector Machine model
    svm_model = SVC(kernel='linear', class_weight="balanced", random_state=42)
    svm_model.fit(X_train, y_train)

    # Predictions for SVM
    y_pred_svm = svm_model.predict(X_test)

    # Evaluation for SVM
    svm_accuracy = accuracy_score(y_test, y_pred_svm)
    print("SVM Accuracy:", svm_accuracy)
    print("SVM Classification Report:\n", classification_report(y_test, y_pred_svm))
```

4. Decision Tree:

```
from sklearn.tree import DecisionTreeClassifier

# Decision Tree Classifier
dt_model = DecisionTreeClassifier(class_weight="balanced", random_state=42)
dt_model.fit(X_train, y_train)

# Predictions for Decision Tree
y_pred_dt = dt_model.predict(X_test)

# Evaluation for Decision Tree
dt_accuracy = accuracy_score(y_test, y_pred_dt)
print("Decision Tree Accuracy:", dt_accuracy)
print("Decision Tree Classification Report:\n", classification_report(y_test, y_pred_dt))
```

5. Gradient Boost Classifier:

```
from sklearn.ensemble import GradientBoostingClassifier

# Gradient Boosting Classifier
gb_model = GradientBoostingClassifier(n_estimators=100, random_state=42)
gb_model.fit(X_train, y_train)

# Predictions for Gradient Boosting
y_pred_gb = gb_model.predict(X_test)

# Evaluation for Gradient Boosting
gb_accuracy = accuracy_score(y_test, y_pred_gb)
print("Gradient Boosting Accuracy:", gb_accuracy)
print("Gradient Boosting Classification Report:\n", classification_report(y_test, y_pred_gb))
```

6. XGBClassifier:

```
[ ] from xgboost import XGBClassifier

    # XGBoost Classifier
    xgb_model = XGBClassifier(use_label_encoder=False, eval_metric="mlogloss", random_state=42)
    xgb_model.fit(X_train, y_train)

    # Predictions for XGBoost
    y_pred_xgb = xgb_model.predict(X_test)

    # Evaluation for XGBoost
    xgb_accuracy = accuracy_score(y_test, y_pred_xgb)
    print("XGBoost Accuracy:", xgb_accuracy)
    print("XGBoost Classification Report:\n", classification_report(y_test, y_pred_xgb))
```

Code of Transformer Model (DistilBERT):

```
[ ]  # ✅ Install dependencies if running in Colab
     !pip install scikit-learn pandas tqdm transformers datasets matplotlib

     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import classification_report, accuracy_score

     # Transformers pipeline
     import torch
     from transformers import DistilBertTokenizerFast, DistilBertForSequenceClassification, Trainer, TrainingArguments
     from datasets import Dataset

     from tqdm import tqdm

     tqdm.pandas()

     # ==== Step 1: Load and Prepare Data ====
     df = pd.read_csv("/content/drive/MyDrive/finalyearproject/UpdatedResumeDataSet.csv")
     print("Dataset Loaded:", df.shape)
```

```
     # Encode labels
     label2id = {label: i for i, label in enumerate(sorted(df["Category"].unique()))}
     id2label = {v: k for k, v in label2id.items()}
     df["label"] = df["Category"].map(label2id)

     # Train-test split
     train_df, test_df = train_test_split(df, test_size=0.2, random_state=42, stratify=df["label"])
     print(" Classes:", len(label2id))

     # ==== Step 2: Fine-tune DistilBERT ====
     print(" Fine-tuning DistilBERT...")
     tokenizer = DistilBertTokenizerFast.from_pretrained("distilbert-base-uncased")

     def tokenize(batch):
         return tokenizer(batch["Resume"], padding=True, truncation=True, max_length=512)
```

```
train_dataset = Dataset.from_pandas(train_df[["Resume", "label"]])
test_dataset = Dataset.from_pandas(test_df[["Resume", "label"]])

train_dataset = train_dataset.map(tokenize, batched=True)
test_dataset = test_dataset.map(tokenize, batched=True)

train_dataset.set_format("torch", columns=["input_ids", "attention_mask", "label"])
test_dataset.set_format("torch", columns=["input_ids", "attention_mask", "label"])

model = DistilBertForSequenceClassification.from_pretrained(
    "distilbert-base-uncased", num_labels=len(label2id)
)

training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    save_strategy="no",
    learning_rate=2e-5,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=5,  # Increased to improve accuracy
    weight_decay=0.01,
    logging_steps=50,
    logging_dir="./logs",
    load_best_model_at_end=False,
    report_to="none",  # Disable wandb
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
)

trainer.train()

# ==== Step 3: Evaluate ====
preds_output = trainer.predict(test_dataset)
bert_preds = np.argmax(preds_output.predictions, axis=1)
bert_acc = accuracy_score(test_df["label"], bert_preds)
```

```
print("BERT Classification Report:")
print(classification_report(test_df["label"], bert_preds, target_names=[id2label[i] for i in sorted(id2label)]))
print("BERT Accuracy:", round(bert_acc * 100, 2), "%")
```

References

1. **5. Vaswani, A., et al. (2017). Attention is All You Need. In Advances in Neural Information Processing Systems (NeurIPS).**
2. **Scikit-learn: Machine Learning in Python – https://scikit-learn.org**
3. **Streamlit: The fastest way to build and share data apps – https://streamlit.io**
4. **Google AI – https://ai.google**
5. **GitHub Repository for LLM APIs – https://github.com/google/generative-ai-python**