Mini Project

About myself:

Hello. My name is G.Nayana Harshitha and I am currently a student pursuing a degree in Computer science engineering. I have a keen interest in developing innovative solutions using emerging technologies,particularly in the fields of AI/ML. This mini project reflects my enthusiasm for learning and applying theoretical concepts to practical scenarios. I am excited to continue growing my skills and contributing to meaningful projects in the tech industry. Thank you for taking the time to review my work!

Objective:

Resume Screening and Ranking System

This project aims to develop an automated resume screening and ranking system using Natural Language Processing (NLP) and machine learning techniques. The system will help recruiters and hiring managers efficiently process large volumes of resumes, identify the most qualified candidates for a given job description, and streamline the overall recruitment process.

By leveraging NLP, the system will extract key information from resumes, such as skills, experience, and education. It will then compare these attributes to a target job description and generate a similarity score, indicating how well a candidate's profile matches the job requirements. The system will also

incorporate additional factors like education level to provide a comprehensive ranking of candidates.

The ultimate goal is to reduce the time and effort spent on manual resume screening while ensuring that the best candidates are identified and prioritized for further consideration.

Code run-through:

1. Data Preprocessing (preprocess_data function):

Objective:

This code focuses on cleaning and preparing the raw resume data for analysis. It addresses common issues in resume data, such as inconsistent formatting and missing information.

Steps:

1. **Data Loading: Loads the raw resume dataset from a CSV file using the Pandas library.**
2. **Column Selection: Selects relevant columns for analysis, such as skills, experience, career objective, and education details.**
3. **Missing Value Handling: Addresses missing values in critical fields. For example, it may drop rows with missing skills or career objectives.**
4. **Skills Normalization: Cleans and standardizes the 'skills' column by removing unnecessary characters and converting it into a list of skills.**
5. **Experience Processing: Extracts and normalizes experience information, if available, to ensure it's in a consistent format.**
6. **Data Saving: Saves the preprocessed dataset to a new CSV file for further use.**

```python
import pandas as pd
import numpy as np

def preprocess_data(input_path, output_path):
    """
    Preprocess the dataset by cleaning and normalizing fields like skills and experience.
    Args:
        input_path (str): Path to the input CSV dataset.
        output_path (str): Path to save the cleaned dataset.
    """
    # Load the dataset
    resumes_df = pd.read_csv(input_path, on_bad_lines='skip')

    # Check if 'experience' column exists, if not, skip processing it
    if 'experience' not in resumes_df.columns:
        print("Warning: 'experience' column not found. Skipping experience processing.")
    else:
        # Select relevant columns
        resumes_df = resumes_df[[
            "skills", "career_objective", "educational_institution_name",
            "degree_names", "major_field_of_studies", "experience"
        ]]

        # ... (rest of your original code for experience processing) ...


    # Drop rows with missing critical fields (outside the 'experience' check)
    resumes_df.dropna(subset=["skills", "career_objective"], inplace=True)

    # Normalize the 'skills' column
    resumes_df["skills"] = resumes_df["skills"].str.strip("[]").str.replace("'", "").str.split(", ")

    # ... (rest of your original code that doesn't depend on 'experience') ...
```

[64]

```python
            "skills", "career_objective", "educational_institution_name",
            "degree_names", "major_field_of_studies", "experience"
        ]]

        # ... (rest of your original code for experience processing) ...


    # Drop rows with missing critical fields (outside the 'experience' check)
    resumes_df.dropna(subset=["skills", "career_objective"], inplace=True)

    # Normalize the 'skills' column
    resumes_df["skills"] = resumes_df["skills"].str.strip("[]").str.replace("'", "").str.split(", '

    # ... (rest of your original code that doesn't depend on 'experience') ...

    # Save the cleaned dataset
    resumes_df.to_csv(output_path, index=False)
    print(f"Cleaned dataset saved to {output_path}")

# Run Preprocessing
input_file = "resume_data.csv"  # Input raw dataset
output_file = "cleaned_resumes.csv"  # Cleaned dataset
preprocess_data(input_file, output_file)
```

2. Embedding and Similarity Calculation (compute_embeddings function):

Objective:

This code transforms textual data (skills and job description) into numerical representations (embeddings)

that can be used to calculate similarity. This process helps the system understand the semantic meaning of words and phrases.

Steps:

1. Model Loading: Loads a pre-trained NLP model (e.g., Sentence Transformer) for embedding generation.
2. Embedding Function: Defines a function to compute embeddings for given text.
3. Job Description Embedding: Computes the embedding for the target job description.
4. Candidate Skills Embeddings: Iterates through the dataset and computes embeddings for each candidate's skills.
5. Similarity Calculation: Uses cosine similarity to measure the similarity between candidate skill embeddings and the job description embedding.
6. Data Augmentation: Adds the computed embeddings and similarity scores as new columns to the dataset.
7. Data Saving: Saves the updated dataset with embeddings to a new CSV file.

```python
from transformers import AutoTokenizer, AutoModel
from sklearn.metrics.pairwise import cosine_similarity

def compute_embeddings(input_path, job_description, output_path):
    """
    Computes embeddings for candidate skills and a given job description.
    Args:
        input_path (str): Path to the cleaned CSV dataset.
        job_description (str): Job description to match candidates.
        output_path (str): Path to save the dataset with embeddings and similarity scores.
    """
    # Load the cleaned dataset
    resumes_df = pd.read_csv(input_path)

    # Load pre-trained NLP model
    model_name = "sentence-transformers/all-MiniLM-L6-v2"
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = AutoModel.from_pretrained(model_name)

    # Define embedding function
    def get_embedding(text):
        inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True, max_length=512)
        outputs = model(**inputs)
        return outputs.last_hidden_state.mean(dim=1).detach().numpy()

    # Compute job description embedding
    job_embedding = get_embedding(job_description)

    # Compute embeddings for candidate skills
    resumes_df["skills_embedding"] = resumes_df["skills"].apply(lambda x: get_embedding(" ".join(eval(x))))

    # Compute similarity scores
    resumes_df["similarity"] = resumes_df["skills_embedding"].apply(lambda x: cosine_similarity(job_embedding, x).flatten()[0])

    # Save the dataset with embeddings
```

```python
    # Compute similarity scores
    resumes_df["similarity"] = resumes_df["skills_embedding"].apply(lambda x: cosine_similarity(job_embedding, x).flatten()[0])

    # Save the dataset with embeddings
    resumes_df.to_csv(output_path, index=False)
    print(f"Embeddings and similarity scores saved to {output_path}")

# Run Embedding and Similarity Calculation
input_file = "cleaned_resumes.csv"
output_file = "resumes_with_embeddings.csv"
job_desc = "Looking for a Python developer with experience in Machine Learning and NLP."
compute_embeddings(input_file, job_desc, output_file)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Co
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 100% 350/350 [00:00<00:00, 16.5kB/s]
vocab.txt: 100% 232k/232k [00:00<00:00, 552kB/s]
tokenizer.json: 100% 466k/466k [00:00<00:00, 9.65MB/s]
special_tokens_map.json: 100% 112/112 [00:00<00:00, 4.31kB/s]
config.json: 100% 612/612 [00:00<00:00, 33.8kB/s]
model.safetensors: 100% 90.9M/90.9M [00:00<00:00, 162MB/s]
Embeddings and similarity scores saved to resumes_with_embeddings.csv
```

3. Candidate Ranking (rank_candidates_with_matched_score function):

Objective:

This code ranks candidates based on their similarity scores, education bonus, and optionally, experience. It assigns a final score to each candidate and sorts them accordingly.

Steps:

1. **Data Loading: Loads the dataset containing the computed embeddings and similarity scores.**
2. **Education Bonus: Assigns an education bonus to candidates based on their educational institution, potentially favoring those with relevant degrees (e.g., B.Tech, M.Tech).**
3. **Final Score Calculation: Calculates a final score for each candidate using a weighted combination of their similarity score and education bonus. Experience can also be included as a factor.**
4. **Candidate Ranking: Sorts candidates in descending order based on their final scores.**
5. **Data Saving: Saves the ranked candidates to a new CSV file.**

```python
def rank_candidates(input_path, output_path, experience_weight=0.2, similarity_weight=0.7, education_bonus=0.1):
    """
    Ranks candidates based on similarity scores, experience, and education bonus.
    Args:
        input_path (str): Path to the dataset with embeddings and similarity scores.
        output_path (str): Path to save the ranked candidates.
    """
    # Load the dataset
    resumes_df = pd.read_csv(input_path)

    # Check for 'experience' column and add it if missing
    if 'experience' not in resumes_df.columns:
        resumes_df['experience'] = 0  # Add a default experience value (e.g., 0)

    # Add education bonus
    resumes_df["education_bonus"] = resumes_df["educational_institution_name"].apply(
        lambda x: education_bonus if isinstance(x, str) and ("B.Tech" in x or "M.Tech" in x) else 0
    )

    # Calculate final score
    resumes_df["final_score"] = (
        resumes_df["similarity"] * similarity_weight +
        resumes_df["experience"] * experience_weight +
        resumes_df["education_bonus"]
    )

    # Rank candidates by final score
    ranked_df = resumes_df.sort_values(by="final_score", ascending=False)

    # Save ranked candidates
    ranked_df.to_csv(output_path, index=False)
    print(f"Ranked candidates saved to {output_path}")
```

```python
import pandas as pd
ranked_df = pd.read_csv("ranked_candidates.csv")
print(ranked_df.columns)
print(ranked_df.head())
```

```
Index(['address', 'career_objective', 'skills', 'educational_institution_name',
       'degree_names', 'passing_years', 'educational_results', 'result_types',
       'major_field_of_studies', 'professional_company_names', 'company_urls',
       'start_dates', 'end_dates', 'related_skils_in_job', 'positions',
       'locations', 'responsibilities', 'extra_curricular_activity_types',
       'extra_curricular_organization_names',
       'extra_curricular_organization_links', 'role_positions', 'languages',
       'proficiency_levels', 'certification_providers', 'certification_skills',
       'online_links', 'issue_dates', 'expiry_dates', 'job_position_name',
       'educational_requirements', 'experiencere_requirement',
       'age_requirement', 'responsibilities.1', 'skills_required',
       'matched_score'],
      dtype='object')
  address                                    career_objective  \
0     NaN  Big data analytics working and database wareho...
1     NaN  Fresher looking to join as a data analyst and ...
2     NaN                                                NaN
3     NaN  To obtain a position in a fast-paced business ...
4     NaN  Professional accountant with an outstanding wo...

                                              skills  \
0  ['Big Data', 'Hadoop', 'Hive', 'Python', 'Mapr...
1  ['Data Analysis', 'Data Analytics', 'Business ...
2  ['Software Development', 'Machine Learning', '...
3  ['accounts payables', 'accounts receivables', ...
4  ['Analytical reasoning', 'Compliance testing k...

                        educational_institution_name  \
0  ['The Amity School of Engineering & Technology...
1  ['Delhi University - Hansraj College', 'Delhi ...
2    ['Birla Institute of Technology (BIT), Ranchi']
3  ['Martinez Adult Education, Business Training ...
4                          ['Kent State University']
```

```
                                        degree_names     passing_years  \
0                                          ['B.Tech']          ['2019']
1    ['B.Sc (Maths)', 'M.Sc (Science) (Statistics)']  ['2015', '2018']
2                                          ['B.Tech']          ['2018']
3  ['Computer Applications Specialist Certificate...          ['2008']
4            ['Bachelor of Business Administration']            [None]

  educational_results    result_types             major_field_of_studies  \
0            ['N/A']          [None]                  ['Electronics']
1      ['N/A', 'N/A']  ['N/A', 'N/A']      ['Mathematics', 'Statistics']
2            ['N/A']         ['N/A']  ['Electronics/Telecommunication']
3             [None]          [None]          ['Computer Applications']
4           ['3.84']          [None]                      ['Accounting']

                         professional_company_names  ... online_links  \
0                                  ['Coca-COla']  ...           NaN
1                              ['BIB Consultancy']  ...           NaN
2                             ['Axis Bank Limited']  ...           NaN
3  ['Company Name ï¼ City , State', 'Company Name...  ...           NaN
4  ['Company Name', 'Company Name', 'Company Name...  ...       [None]

  issue_dates           expiry_dates  \
0         NaN                    NaN
1         NaN                    NaN
2         NaN                    NaN
3         NaN                    NaN
4      [None]  ['February 15, 2021']

                                job_position_name  \
0                          Senior Software Engineer
1                      Machine Learning (ML) Engineer
2  Executive/ Senior Executive- Trade Marketing, ...
3                      Business Development Executive
4                                Senior iOS Engineer
```

```
                          educational_requirements experiencere_requirement  \
0  B.Sc in Computer Science & Engineering from a ...          At least 1 year
1  M.Sc in Computer Science & Engineering or in a...       At least 5 year(s)
2            Master of Business Administration (MBA)         At least 3 years
3                                  Bachelor/Honors             1 to 3 years
4      Bachelor of Science (BSc) in Computer Science         At least 4 years

      age_requirement                                responsibilities.1  \
0                 NaN  Technical Support\nTroubleshooting\nCollaborat...
1                 NaN  Machine Learning Leadership\nCross-Functional ...
2                 NaN  Trade Marketing Executive\nBrand Visibility, S...
3  Age 22 to 30 years  Apparel Sourcing\nQuality Garment Sourcing\nRe...
4                 NaN  iOS Lifecycle\nRequirement Analysis\nNative Fr...

                                   skills_required matched_score
0                                              NaN      0.850000
1                                              NaN      0.750000
2  Brand Promotion\nCampaign Management\nField Su...      0.416667
3  Fast typing skill\nIELTSInternet browsing & on...      0.760000
4  iOS\niOS App Developer\niOS Application Develo...      0.650000

[5 rows x 35 columns]
```

4. Visualization (visualize_rankings function):

Objective:

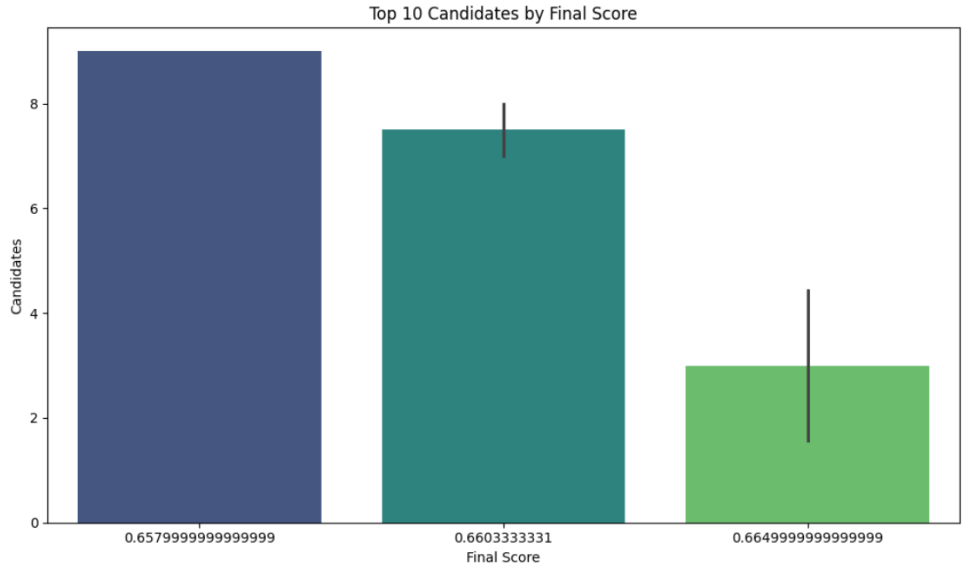
These codes provides visualizations to present the ranking results and insights more effectively. This helps users understand the candidate distribution and identify top contenders quickly.

Steps:

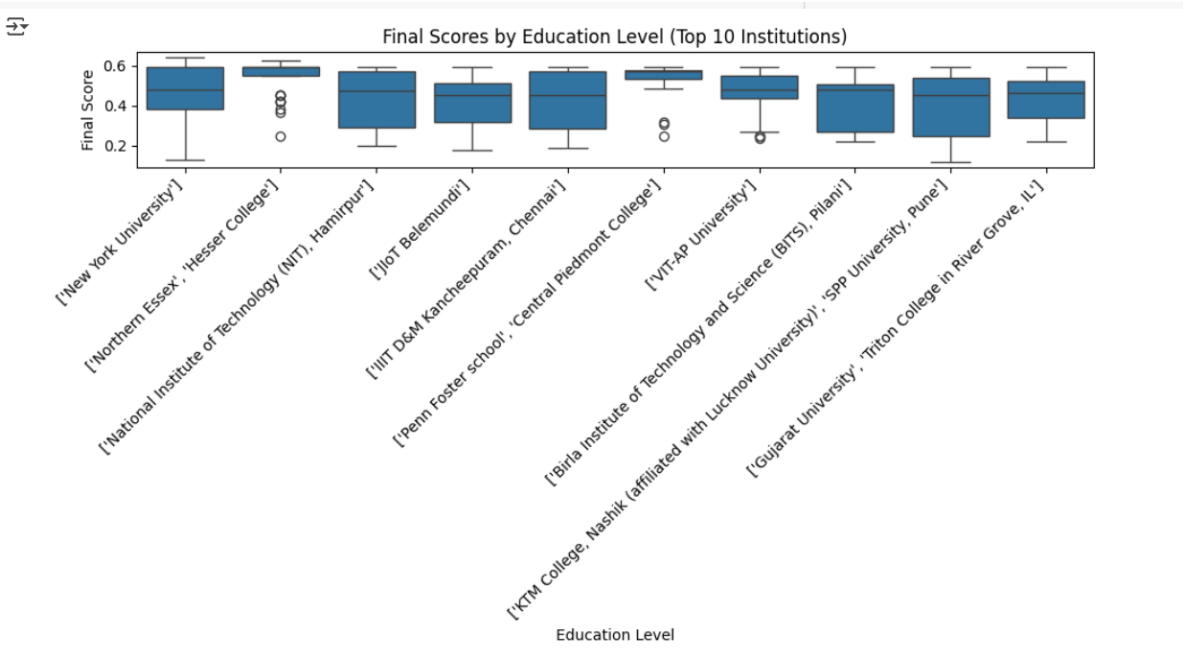
1. **Data Loading: Loads the ranked candidates data.**
2. **Bar Plot: Creates a bar plot to display the top 10 candidates by final score.**
3. **Distribution Plot: Creates a histogram to show the distribution of final scores among all candidates.**
4. **Box Plot: Creates a box plot to compare the distribution of final scores across different education levels, focusing on the top 10 most frequent institutions.**

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

def visualize_rankings(input_path):
    """
    Visualizes the top-ranked candidates.
    Args:
        input_path (str): Path to the ranked candidates CSV file.
    """
    # Load the ranked candidates dataset
    ranked_df = pd.read_csv(input_path)

    # Select top candidates
    top_candidates = ranked_df.head(10)

    # Plot final scores
    plt.figure(figsize=(10, 6))
    sns.barplot(data=top_candidates, x="final_score", y=top_candidates.index, palette="viridis")
    plt.title("Top 10 Candidates by Final Score")
    plt.xlabel("Final Score")
    plt.ylabel("Candidates")
    plt.tight_layout()
    plt.show()

# Run Visualization
visualize_rankings("ranked_candidates.csv")
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the s

  sns.barplot(data=top_candidates, x="final_score", y=top_candidates.index, palette="viridis")
```



```python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

def visualize_rankings(input_path):
    """
    Visualizes the top-ranked candidates using various plots.
    Args:
        input_path (str): Path to the ranked candidates CSV file.
    """
    # Load the ranked candidates dataset
    ranked_df = pd.read_csv(input_path)

    # ... (other visualizations remain the same) ...

    # 3. Box plot of final scores by education level (if available)
    if "educational_institution_name" in ranked_df.columns:
        plt.figure(figsize=(10, 6))

        # Limit the number of institutions displayed (e.g., top 10)
        top_institutions = ranked_df["educational_institution_name"].value_counts().head(10).index
        filtered_df = ranked_df[ranked_df["educational_institution_name"].isin(top_institutions)]

        sns.boxplot(data=filtered_df, x="educational_institution_name", y="final_score")
        plt.title("Final Scores by Education Level (Top 10 Institutions)")
        plt.xlabel("Education Level")
        plt.ylabel("Final Score")

        # Rotate x-axis labels for better readability
        plt.xticks(rotation=45, ha="right")
        plt.tight_layout()
        plt.show()

# Run Visualization
visualize_rankings("ranked_candidates.csv")
```



```python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

def visualize_rankings(input_path):
    """
    Visualizes the top-ranked candidates using various plots.
    Args:
        input_path (str): Path to the ranked candidates CSV file.
    """
    # Load the ranked candidates dataset
    ranked_df = pd.read_csv(input_path)

    # ... (other visualizations remain the same) ...

    # 2. Distribution of final scores
    plt.figure(figsize=(10, 6))
    sns.histplot(ranked_df["final_score"], bins=20, kde=True)
    plt.title("Distribution of Final Scores")
    plt.xlabel("Final Score")
    plt.ylabel("Frequency")
    plt.tight_layout()

    # Save the distribution plot to a file
    plt.savefig("final_score_distribution.png")  # You can change the file name and format

    plt.show()  # Still show the plot in the notebook

    # ... (rest of the visualization code) ...

# Run Visualization
visualize_rankings("ranked_candidates.csv")
```