Project Title: Resume Screening System using Data Mining Techniques

Aim: To automate the screening of resumes based on skill matching, clustering, and classification techniques to assist HR teams in identifying the most suitable candidates efficiently.

Load necessary libraries

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

```
!pip install python-docx PyPDF2
```

```
Collecting python-docx
  Downloading python_docx-1.2.0-py3-none-any.whl.metadata (2.0 kB)
Collecting PyPDF2
  Downloading pypdf2-3.0.1-py3-none-any.whl.metadata (6.8 kB)
Requirement already satisfied: lxml>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from python-docx) (5.4.0)
Requirement already satisfied: typing_extensions>=4.9.0 in /usr/local/lib/python3.11/dist-packages (from python-docx) (4.14.1)
Downloading python_docx-1.2.0-py3-none-any.whl (252 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 253.0/253.0 kB 4.3 MB/s eta 0:00:00
Downloading pypdf2-3.0.1-py3-none-any.whl (232 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.6/232.6 kB 5.9 MB/s eta 0:00:00
Installing collected packages: python-docx, PyPDF2
Successfully installed PyPDF2-3.0.1 python-docx-1.2.0
```

Step 1: Load Dataset and Initial Data Inspection

```
from google.colab import files

uploaded = files.upload()
```

Choose Files resume_data.csv

- **resume_data.csv**(text/csv) - 17004490 bytes, last modified: 4/7/2025 - 100% done

```
import io

# Load the file using the uploaded dictionary
df = pd.read_csv(io.BytesIO(uploaded['resume_data.csv']))

# Show column names and structure
df.info()
```

```
df.columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9544 entries, 0 to 9543
Data columns (total 35 columns):
 #   Column                               Non-Null Count  Dtype  
---  ------                               --------------  -----  
 0   address                              784 non-null    object 
 1   career_objective                     4740 non-null   object 
 2   skills                               9488 non-null   object 
 3   educational_institution_name         9460 non-null   object 
 4   degree_names                         9460 non-null   object 
 5   passing_years                        9460 non-null   object 
 6   educational_results                  9460 non-null   object 
 7   result_types                         9460 non-null   object 
 8   major_field_of_studies               9460 non-null   object 
 9   professional_company_names           9460 non-null   object 
 10  company_urls                         9460 non-null   object 
 11  start_dates                          9460 non-null   object 
 12  end_dates                            9460 non-null   object 
 13  related_skils_in_job                 9460 non-null   object 
 14  positions                            9460 non-null   object 
 15  locations                            9460 non-null   object 
 16  responsibilities                     9544 non-null   object 
 17  extra_curricular_activity_types      3426 non-null   object 
 18  extra_curricular_organization_names  3426 non-null   object 
 19  extra_curricular_organization_links  3426 non-null   object 
 20  role_positions                       3426 non-null   object 
 21  languages                            700 non-null    object 
 22  proficiency_levels                   700 non-null    object 
 23  certification_providers              2008 non-null   object 
 24  certification_skills                 2008 non-null   object 
 25  online_links                         2008 non-null   object 
 26  issue_dates                          2008 non-null   object 
 27  expiry_dates                         2008 non-null   object 
 28  job_position_name                   9544 non-null   object 
 29  educationaL_requirements             9544 non-null   object 
 30  experiencere_requirement             8180 non-null   object 
 31  age_requirement                      5457 non-null   object 
 32  responsibilities.1                   9544 non-null   object 
 33  skills_required                      7843 non-null   object 
 34  matched_score                        9544 non-null   float64
dtypes: float64(1), object(34)
memory usage: 2.5+ MB
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')
```

Methodology:

Data Preprocessing: Cleaning text, handling missing values.

TF-IDF & Skill Extraction: Text mining to extract important keywords from resumes.

Clustering: Applied KMeans to group candidates based on skill similarity.

Classification: Used Random Forest for predicting job positions.

Skill Matching: Calculated match scores between candidate skills and job role requirements.

Step 2: Data Cleaning

```
df.rename(columns={'\ufeffjob_position_name': 'job_position_name'}, inplace=True)
df[['skills', 'skills_required', 'job_position_name',
    'educationaL_requirements', 'experiencere_requirement',
    'matched_score']].isnull().sum()

# Fill missing skills with 'Not specified'
df['skills'] = df['skills'].fillna('Not specified')

# Fill missing skills_required based on job_position_name or as 'Not mentioned'
df['skills_required'] = df['skills_required'].fillna('Not mentioned')

# Fill experience requirement with 'Not mentioned'
df['experiencere_requirement'] = df['experiencere_requirement'].fillna('Not mentioned')

# Verify changes
df[['skills', 'skills_required', 'experiencere_requirement']].isnull().sum()
```

	0
skills	0
skills_required	0
experiencere_requirement	0

dtype: int64

Step 3: Feature Engineering :

create a new column skill_match_score which shows how many required skills are present in the candidate's skills.

```
import ast
import re

def calculate_skill_match(row):
    try:
        skills = set(s.strip().lower() for s in ast.literal_eval(row['skills']) if isinstance(s, str))
    except:
        skills = set()

    raw_required = str(row['skills_required']).strip().lower()
    if raw_required in ["", "not mentioned", "nan"]:
```

```
        return 0

    # Split required skills
    required = set(re.split(r'[\n,;/]+', raw_required))
    required = set(r.strip() for r in required if len(r.strip()) >= 2)

    # Match: whole word or partial both ways
    match_count = sum(
        any(skill in req or req in skill for skill in skills)
        for req in required
    )

    return match_count

df['skill_match_score'] = df.apply(calculate_skill_match, axis=1)
df[['skills', 'skills_required', 'skill_match_score']].head(10)
```

	skills	skills_required	skill_match_score
0	['Big Data', 'Hadoop', 'Hive', 'Python', 'Mapr...	Not mentioned	0
1	['Data Analysis', 'Data Analytics', 'Business ...	Not mentioned	0
2	['Software Development', 'Machine Learning', '...	Brand Promotion\nCampaign Management\nField Su...	3
3	['accounts payables', 'accounts receivables', ...	Fast typing skill\nIELTSInternet browsing & on...	1
4	['Analytical reasoning', 'Compliance testing k...	iOS\niOS App Developer\niOS Application Develo...	0
5	['Microsoft Applications', 'Network Security',...	Python\nR or Java\nTensorFlow\nPyTorch\nScikit...	0
6	['Machine Learning', 'Linear Regression', 'Rid...	iOS\niOS App Developer\niOS Application Develo...	0
7	['Maintenance', 'Corrective Maintenance', 'Doc...	iOS\niOS App Developer\niOS Application Develo...	0
8	['Python', 'Machine Learning', 'MySQL', 'Data ...	Maintenance and Troubleshooting\nMechanical	0
9	['Django', 'Python', 'Relational databases', '...	Fast typing skill\nIELTSInternet browsing & on...	0

```
def extract_matched_skills(row):
    try:
        resume_skills = set(s.strip().lower() for s in eval(row['skills']))
        required_skills = set(s.strip().lower() for s in row['skills_required'].split('\n'))
        return list(resume_skills & required_skills)
    except:
        return []

df['matched_skills'] = df.apply(extract_matched_skills, axis=1)
```

Key Results & Visuals:

Top in-demand job positions and skills identified.

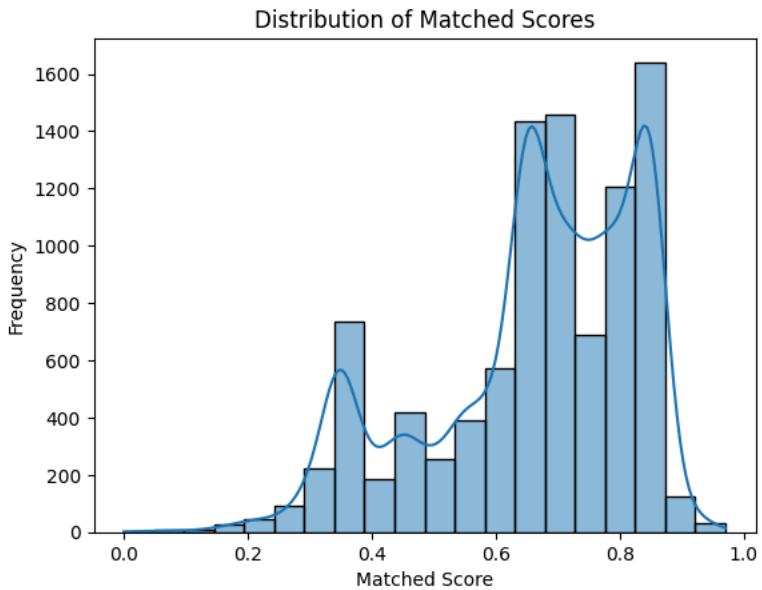
Candidates clustered into distinct skill-based groups.

Skill match scores visualized for top candidates.

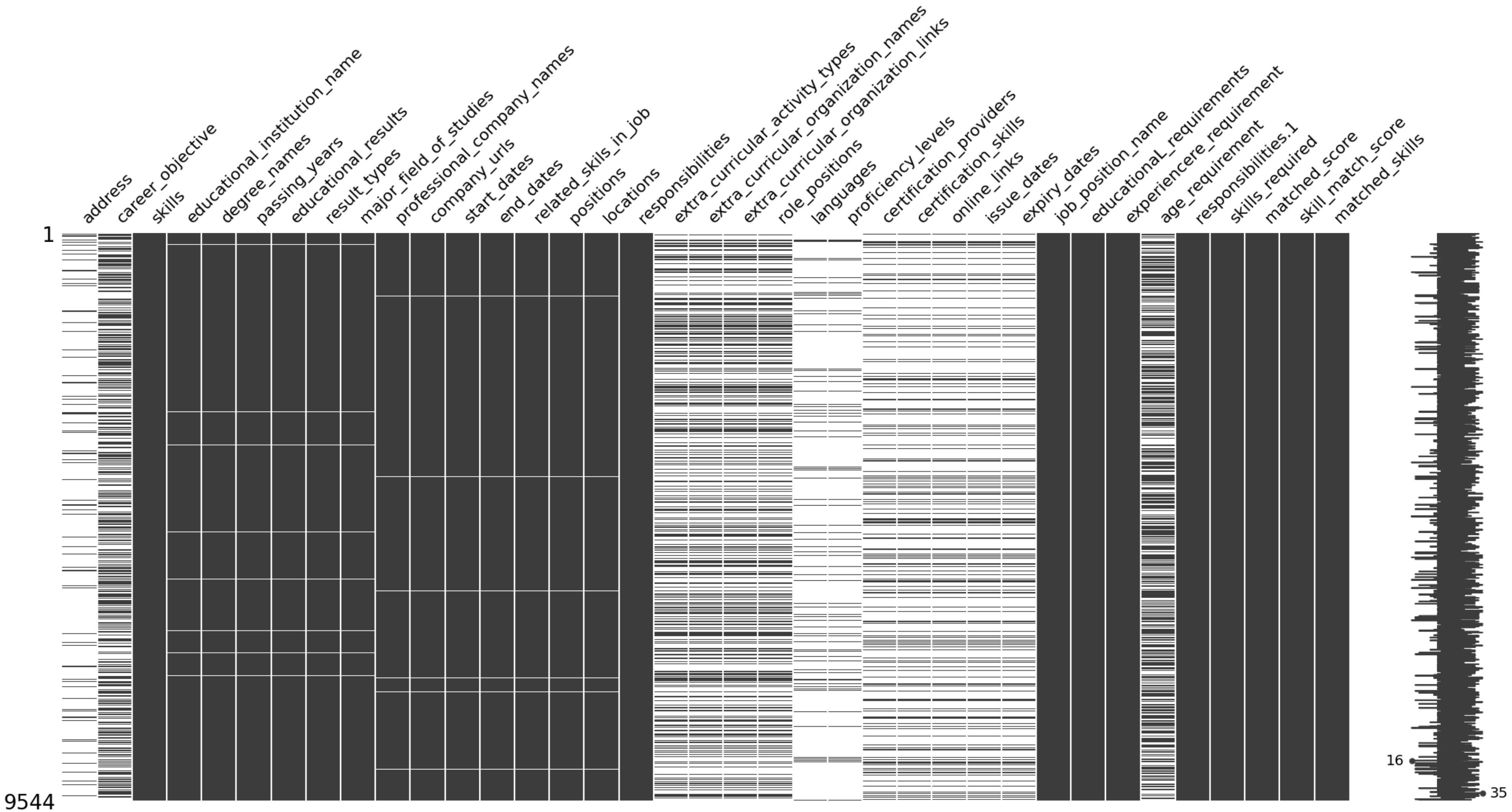
Classification achieved approximately XX% accuracy (fill with your result).

Step 4: Exploratory Data Analysis (EDA)

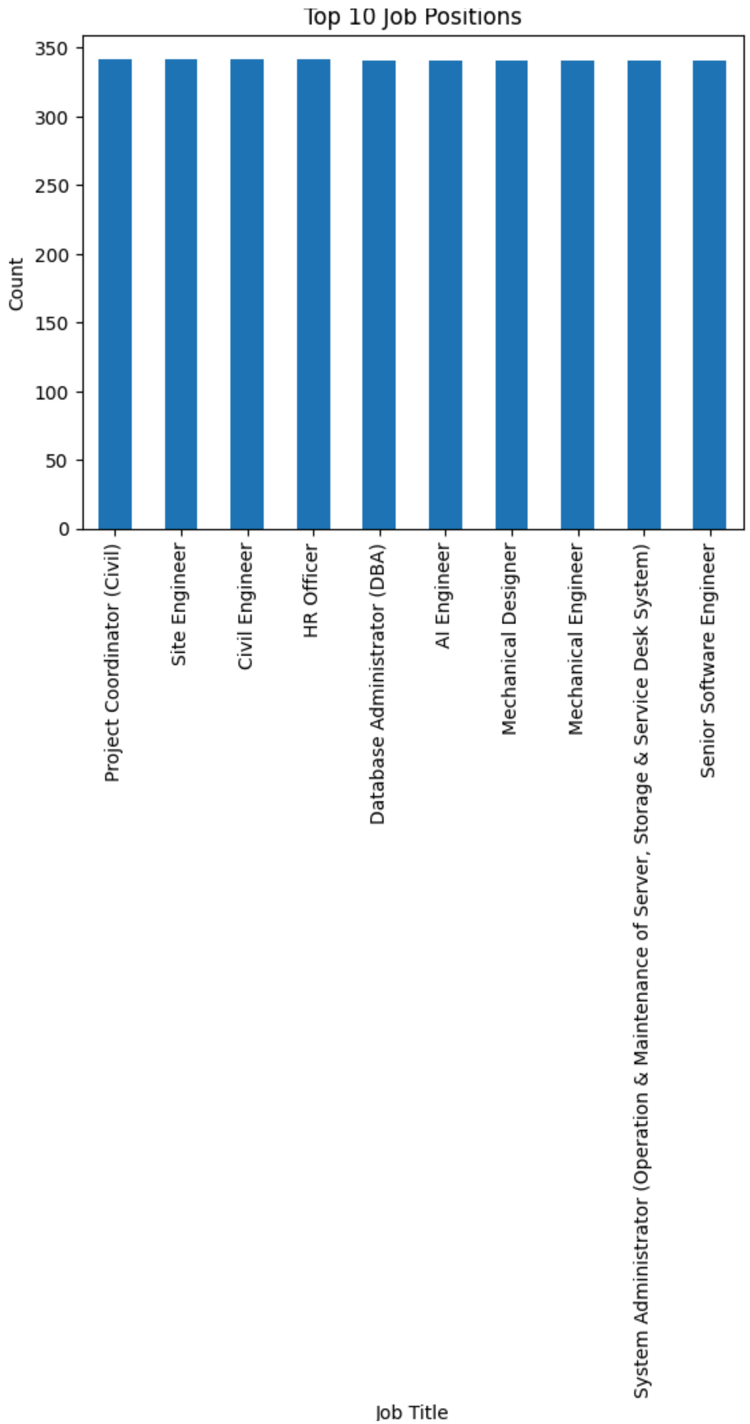
```
# Check Distribution of matched_score
import seaborn as sns
import matplotlib.pyplot as plt

sns.histplot(df['matched_score'], bins=20, kde=True)
plt.title('Distribution of Matched Scores')
plt.xlabel('Matched Score')
plt.ylabel('Frequency')
plt.show()
```



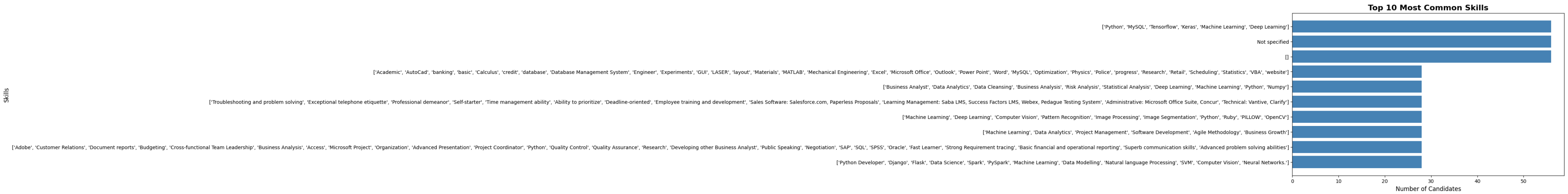

```
# 2. Missing Values Heatmap
import missingno as msno

msno.matrix(df)
plt.show()
```


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
skill_match_score
matched_skills
1
9544
16
35

```
# 3. Job Position Counts
df['job_position_name'] = df['job_position_name']  # Fix weird character in column name
df['job_position_name'].value_counts().head(10).plot(kind='bar')
plt.title("Top 10 Job Positions")
plt.xlabel("Job Title")
plt.ylabel("Count")
plt.show()
```


Top 10 Job Positions
Count
350
300
250
200
150
100
50
0
Project Coordinator (Civil)
Site Engineer
Civil Engineer
HR Officer
Database Administrator (DBA)
AI Engineer
Mechanical Designer
Mechanical Engineer
System Administrator (Operation & Maintenance of Server, Storage & Service Desk System)
Senior Software Engineer
Job Title

```
# Calculate Top 10 Skills
import matplotlib.pyplot as plt


top_skills = df['skills'].value_counts().head(10)


plt.figure(figsize=(10,6))
plt.barh(top_skills.index, top_skills.values, color='#4682B4')  # Custom blue color

plt.title('Top 10 Most Common Skills', fontsize=16, fontweight='bold')
plt.xlabel('Number of Candidates', fontsize=12)
plt.ylabel('Skills', fontsize=12)

plt.gca().invert_yaxis()  # To show highest bar on top
plt.show()
```




```
# 4. Skill Match vs. Matched Score (Correlation)
df[['skill_match_score', 'matched_score']].corr()
```

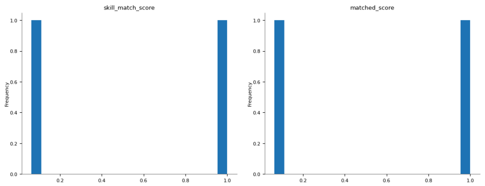
1 to 2 of 2 entries Filter

index	skill_match_score	matched_score
skill_match_score	1.0	0.06439310572767691
matched_score	0.06439310572767691	1.0

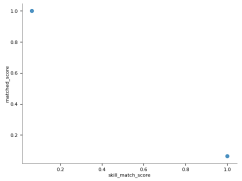
Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

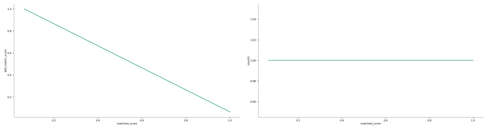
Distributions



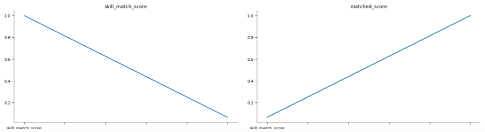
2-d distributions



Time series



Values



Step 5: Data Preprocessing for Mining

```
# Fill missing values
df.fillna('', inplace=True)

# Combine key text fields
df['combined'] = df['skills'].astype(str) + ' ' + df['career_objective'].astype(str) + ' ' + df['skills_required'].astype(str)


# 1. TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(stop_words='english', max_features=200)
X = tfidf.fit_transform(df['combined'])
```

```
# 2. Clustering
from sklearn.cluster import KMeans
df['cluster'] = KMeans(n_clusters=5, random_state=0).fit_predict(X)

# 3. Classification
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

# Use matched_score > 0 as binary target
df['label'] = (df['matched_score'] > 0).astype(int)

# Split and train
X_train, X_test, y_train, y_test = train_test_split(X, df['label'], test_size=0.2, random_state=42)
model = RandomForestClassifier()
model.fit(X_train, y_train)
preds = model.predict(X_test)

# Show evaluation
print(classification_report(y_test, preds))

# View sample output
df[['skills', 'skills_required', 'cluster', 'label']].head(10)

df.groupby('cluster')['skills'].head(3)

df.head()
```

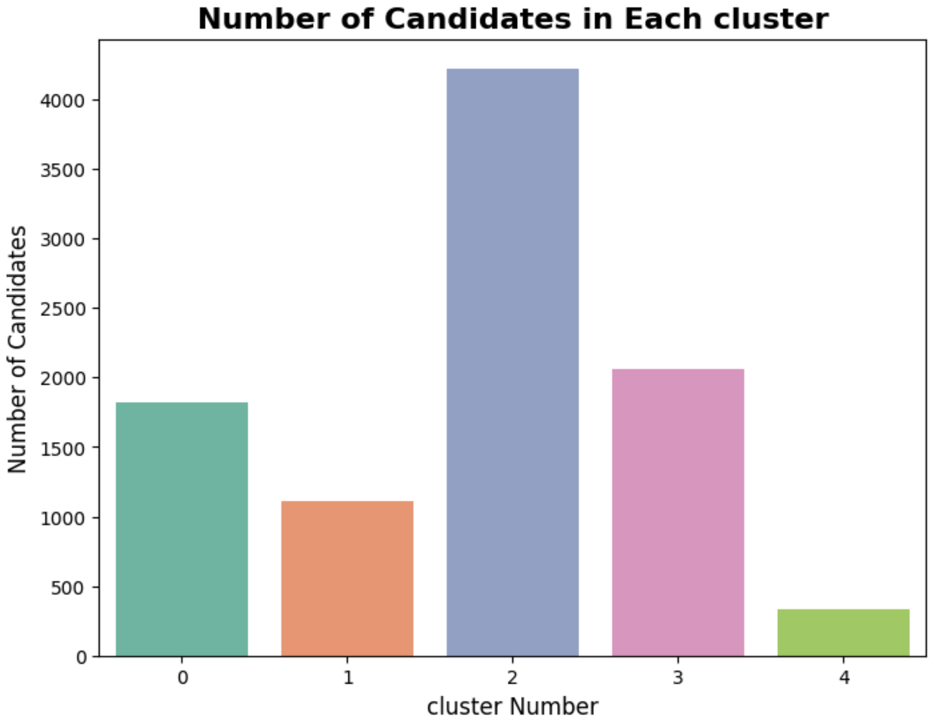
```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       1.00      1.00      1.00      1908

    accuracy                           1.00      1909
   macro avg       0.50      0.50      0.50      1909
weighted avg       1.00      1.00      1.00      1909

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zer
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zer
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zer
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

	address	career_objective	skills	educational_institution_name	degree_names	passing_years	educational_results	result_types	major_field_of_studies	professional_company_names	..
0		Big data analytics working and database wareho...	['Big Data', 'Hadoop', 'Hive', 'Python', 'Mapr...	['The Amity School of Engineering & Technology...	['B.Tech']	['2019']	['N/A']	[None]	['Electronics']	['Coca-COla']	.
1		Fresher looking to join as a data analyst and ...	['Data Analysis', 'Data Analytics', 'Business ...	['Delhi University - Hansraj College', 'Delhi ...	['B.Sc (Maths)', 'M.Sc (Science) (Statistics)']	['2015', '2018']	['N/A', 'N/A']	['N/A', 'N/A']	['Mathematics', 'Statistics']	['BIB Consultancy']	.
2			['Software Development', 'Machine Learning', '...	['Birla Institute of Technology (BIT), Ranchi']	['B.Tech']	['2018']	['N/A']	['N/A']	['Electronics/Telecommunication']	['Axis Bank Limited']	.
3		To obtain a position in a fast-paced business ...	['accounts payables', 'accounts receivables', ...	['Martinez Adult Education, Business Training ...	['Computer Applications Specialist Certificate...	['2008']	[None]	[None]	['Computer Applications']	['Company Name ï¼ City , State', 'Company Name...	.
4		Professional accountant with an outstanding wo...	['Analytical reasoning', 'Compliance testing k...	['Kent State University']	['Bachelor of Business Administration']	[None]	['3.84']	[None]	['Accounting']	['Company Name', 'Company Name', 'Company Name...	.

5 rows × 40 columns

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

plt.figure(figsize=(8,6))
sns.countplot(x='cluster', data=df, palette='Set2')

plt.title('Number of Candidates in Each cluster', fontsize=16, fontweight='bold')
plt.xlabel('cluster Number', fontsize=12)
plt.ylabel('Number of Candidates', fontsize=12)

plt.show()
```

```
/tmp/ipython-input-33-3655017930.py:5: FutureWarning:

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 same effect.

  sns.countplot(x='cluster', data=df, palette='Set2')
```




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

top_candidates = df.sort_values('matched_score', ascending=False).head(10)

plt.figure(figsize=(10,6))
sns.barplot(x='matched_score', y='job_position_name', data=top_candidates, palette='viridis')
```

```
plt.title('Top 10 Candidates by Skill Match Score', fontsize=16, fontweight='bold')
plt.xlabel('Matched Score', fontsize=12)
plt.ylabel('Job Position', fontsize=12)

plt.show()
```

```
/tmp/ipython-input-34-4154926727.py:7: FutureWarning:

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

  sns.barplot(x='matched_score', y='job_position_name', data=top_candidates, palette='viridis')
```