Jaypee Institute of Information Technology, Noida

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING AND INFORMATION TECHNOLOGY

Project Title:

JOB_PULSE - A Job Recommendation Model

Enrolment No.	Name of Student
9921103014	Suyash Rai
9921103043	Divya Darshana

Course Name: Introduction to Big Data & Data Analytics

Course Code: 20B12CS333

Program: B. Tech. CS&E

3rd Year 5th Sem

2023- 2024

TABLE OF CONTENTS

1. ABSTRACT

This study introduces a Job Recommendation Model that leverages advanced data analytics to optimize the job search experience. By combining content-based filtering for predicting relevant job postings based on user queries and collaborative filtering for recommending jobs to users with similar preferences, the model achieves a dual-purpose system.

The use of Natural Language Processing (NLP) ensures nuanced content analysis for accurate predictions, while collaborative filtering fosters a sense of community and shared job-seeking experiences. Significantly, this model not only assists users in discovering tailored job opportunities but also promotes collaboration among users with similar career aspirations, creating a supportive and engaging job-seeking environment.

Evaluation metrics demonstrate the model's effectiveness in providing personalized recommendations, highlighting its potential to transform traditional job search paradigms and enhance user satisfaction. This research marks a significant step forward in developing comprehensive and adaptive systems that better align with individual preferences and foster collaborative job discovery.

2. INTRODUCTION:

In the ever-evolving landscape of job search and recruitment, traditional methods often fall short in providing personalized and collaborative experiences. This project introduces an Enhanced Job Recommendation Model designed to address these challenges by combining advanced analytics and collaborative filtering techniques. The model aims to revolutionize job matching, offering both personalized job suggestions based on individual preferences and collaborative recommendations by connecting users with similar career aspirations.

2.1. PROBLEM STATEMENT:

Traditional job recommendation systems often lack the ability to provide tailored suggestions, leading to a disconnect between job seekers and relevant opportunities. The absence of collaboration features further limits the community aspect of job searching. This project seeks to tackle these issues by developing an enhanced model that not only predicts job matches based on user preferences but also fosters collaboration among users with similar career goals.

2.2. MOTIVATION:

The motivation behind this project stems from the shortcomings of existing job recommendation systems. Many current models lack the finesse needed to deliver personalized suggestions, resulting in job seekers being inundated with irrelevant opportunities. The desire for a more inclusive and collaborative job search experience drives this project. By leveraging advanced analytics and collaborative filtering, we aim to create a model that not only understands individual preferences but also facilitates connections among users sharing similar career trajectories.

This project's ultimate goal is to enhance the job search journey, offering a dynamic and supportive environment where users receive recommendations tailored to their unique preferences while also benefiting from insights shared by a collaborative job-seeking community.

2.3. OBJECTIVES:

Develop a Machine Learning Model: Engineer an advanced machine learning model to optimize the job recommendation process. The primary objective is to create an intelligent system capable of accurately suggesting job opportunities based on users' individual preferences, historical interactions, and relevant data points.

Personalized Job Matching: Implement a personalized job matching algorithm within the model. This feature will analyze user-specific data, including job history, skills, and preferences, to deliver tailored job recommendations that align with individual career aspirations.

Integrate Collaborative Filtering: Incorporate collaborative filtering techniques to enhance the model's recommendations. By identifying patterns and similarities among users with comparable career trajectories, the system aims to foster a sense of community and shared experiences, providing a more nuanced and collaborative job search experience.

Enhance User Satisfaction: Ultimately, the overarching goal is to elevate user satisfaction by delivering a sophisticated, personalized, and collaborative job-matching experience. The model aims to transcend traditional recommendations, providing users with a dynamic platform that caters to their unique professional journey while fostering meaningful connections within the job-seeking community.

2.4. CONTRIBUTION:

The Enhanced Job Recommendation Model makes a substantial contribution to the job search landscape by introducing an automated mechanism that simplifies and accelerates the decision-making process. The primary contribution lies in the creation of a more efficient and personalized job recommendation system, reducing the reliance on manual effort, improving time efficiency, and enhancing cost-effectiveness. This model offers a scalable solution for evaluating job compatibility, ultimately providing users with a streamlined and dynamic platform for discovering relevant opportunities tailored to their unique preferences.

3.1 PROPOSED WORK WITH TOOLS AND DATASETS USED

Traditional rule-based Prediction systems struggle to keep up with evolving Criteria Analysis, leading to a shift towards Machine Learning (ML).

Key advancements include:

Data inspecting

Data inspecting involves evaluating and understanding the characteristics of the dataset, ensuring data quality, and identifying patterns, outliers, or biases. This critical step enhances model performance by informing preprocessing decisions, mitigating biases, and ensuring the robustness of the learning process.

Data imputation

Data imputation in machine learning involves filling missing values within a dataset, ensuring completeness for accurate model training. It enhances the robustness of models by maintaining the integrity of the input features and improving overall predictive performance.

Feature engineering

Extract meaningful features from job descriptions, titles, user profiles, and historical interactions to provide the model with relevant information for accurate recommendations.

Natural Language Processing (NLP) technique is used to analyse and understand the content of job postings and user profiles.

Implementation of Machine Learning Techniques

Implement content and collaborative filtering algorithms to identify patterns and similarities among users, enabling the model to offer personalized job recommendations based on shared preferences.

About the datasets

The datasets used to train and test the ML model for this project is sourced from Kaggle

https://www.kaggle.com/c/job-recommendation/data

It contains the following datasets:

- **users.tsv** contains information about the users. Each row of this file describes a user. The UserID column contains a user's unique id number and the Split column tells whether the user is in the Train group or the Test group. The remaining columns contain demographic and professional information about the users.

```
: users.head(1)
```

	UserID	WindowID	Split	City	State	Country	ZipCode	DegreeType	Major	GraduationDate	WorkHistoryCount	TotalYearsExperience	CurrentlyEmploy
0	47	1	Train	Paramount	CA	US	90723	High School	NaN	1999-06-01 00:00:00	3	10.0	Y

```
: users.columns

: Index(['UserID', 'WindowID', 'Split', 'City', 'State', 'Country', 'ZipCode',
        'DegreeType', 'Major', 'GraduationDate', 'WorkHistoryCount',
        'TotalYearsExperience', 'CurrentlyEmployed', 'ManagedOthers',
        'ManagedHowMany'],
       dtype='object')
```

```
: users.shape

: (389708, 15)
```

DDDDDDDDDDDDDDDD

- **test_users.tsv** contains a list of the Test UserIDs and windows, for our convenience. All of the information in this file can be found in users.tsv.

```
: test_users.head()
```

	UserID	WindowID
0	767	1
1	769	1
2	861	1
3	1006	1
4	1192	1

```
: test_users.columns

: Index(['UserID', 'WindowID'], dtype='object')
```

```
: test_users.shape

: (22838, 2)
```

- **user_history.tsv** contains information about a user's work history. Each row of this file describes a job that a user held. The UserID and Split columns have the same meaning as before. The JobTitle column represents the title of the job, and the Sequence column represents the order in which the user held that job, with smaller numbers indicating more recent jobs.

```
user_history.head()
```

	UserID	WindowID	Split	Sequence	JobTitle
0	47	1	Train	1	National Space Communication Programs-Special ...
1	47	1	Train	2	Detention Officer
2	47	1	Train	3	Passenger Screener, TSA
3	72	1	Train	1	Lecturer, Department of Anthropology
4	72	1	Train	2	Student Assistant

```
user_history.columns
```

```
Index(['UserID', 'WindowID', 'Split', 'Sequence', 'JobTitle'], dtype='object')
```

```
user_history.shape
```

```
(1753901, 5)
```

- **jobs.tsv** contains information about job postings. Each row of this file describes a job post. The JobID column contains the job posting's unique id number.The other columns contain information about the job posting. Two of these columns deserve special attention, the StartDate and EndDate columns. These indicate the period in which this job posting was visible on careerbuilder.com

```
jobs.head(1)
```

	JobID	WindowID	Title	Description	Requirements	City	State	Country	Zip5	StartDate	EndDate
0	1	1	Security Engineer/Technical Lead	<p>Security Clearance Required: Top Secr...	<p>SKILL SET</p>\r<p> </p>\r<p>Network Se...	Washington	DC	US	20531	2012-03-07 13:17:01.643	2012-04-06 23:59:59

```
jobs.columns
```

```
Index(['JobID', 'WindowID', 'Title', 'Description', 'Requirements', 'City',
       'State', 'Country', 'Zip5', 'StartDate', 'EndDate'],
      dtype='object')
```

```
jobs.shape
```

```
(96988, 11)
```

- **apps.tsv** contains information about applications made by users to jobs. Each row describes an application. The UserID, Split, and JobID columns have the same meanings as above, and the ApplicationDate column indicates the date and time at which UserID applied to JobId.

```
apps.head()
```

	UserID	WindowID	Split	ApplicationDate	JobID
0	47	1	Train	2012-04-04 15:56:23.537	169528
1	47	1	Train	2012-04-06 01:03:00.003	284009
2	47	1	Train	2012-04-05 02:40:27.753	2121
3	47	1	Train	2012-04-05 02:37:02.673	848187
4	47	1	Train	2012-04-05 22:44:06.653	733748

```
apps.columns
```

```
Index(['UserID', 'WindowID', 'Split', 'ApplicationDate', 'JobID'], dtype='object')
```

```
apps.shape
```

```
(1603111, 5)
```

2.4. ALGORITHM

Introduction to TF-IDF and Cosine Similarity:

In the realm of natural language processing and information retrieval, TF-IDF (Term Frequency-Inverse Document Frequency) and Cosine Similarity play pivotal roles. TF-IDF is a numerical statistic that reflects the importance of a term in a document relative to a collection, while Cosine Similarity measures the cosine of the angle between two non-zero vectors. Together, they form a powerful combination for extracting meaningful insights and patterns from textual data.

Algorithm Overview - TF-IDF:

TF-IDF transforms a collection of text documents into numerical vectors, considering the importance of terms within each document and across the entire corpus. The algorithm calculates a weight for each term based on its frequency in a document (TF) and its rarity in the entire collection (IDF). The resulting TF-IDF vectors represent the unique features of each document, capturing the significance of terms in the context of the entire dataset.

Key Features of TF-IDF:

1. Term Importance: TF-IDF assigns higher weights to terms that are frequent in a document but rare in the entire corpus, emphasizing their importance in characterizing the document's content.

2. Vector Representation: Documents are transformed into numerical vectors, enabling efficient computation and comparison of textual data.

3. Document Relevance: TF-IDF aids in identifying the relevance of terms within each document, contributing to more accurate information retrieval.

Algorithm Overview - Cosine Similarity:

Cosine Similarity measures the cosine of the angle between two non-zero vectors, providing a numerical value indicating the similarity between the vectors. In the context of TF-IDF, cosine similarity is often employed to assess the similarity between documents based on their TF-IDF vector representations.

Key Features of Cosine Similarity:

1. Similarity Measurement: Cosine Similarity quantifies the similarity between two documents, where a higher value indicates greater similarity.

2. Vector Space Model: Utilizes the vector space model to represent documents, facilitating efficient comparisons and similarity assessments.

3. Angle-Based Metric: The metric is based on the angle between vectors, making it robust against differences in document lengths.

Together, TF-IDF and Cosine Similarity form a powerful duo for extracting meaningful information, enabling document comparison, and supporting tasks such as information retrieval, recommendation systems, and clustering. Their key features contribute to their versatility in handling diverse textual datasets.

3.2. GENERAL WORKFLOW

1. Dataset Splitting:

- [Dataset] → [Training Set] + [Testing Set]

2. Model Training:

- [Training Set] → [Train Recommendation Model]

3. Model Evaluation:

- [Train Recommendation Model] + [Testing Set] → [Generate Recommendations]

4. Generate Recommendations:

- [Generate Recommendations] → [Recommended Items]

5. Evaluate Recommendations:

- [Testing Set] + [Recommended Items] → [Evaluation Metrics]

6. Fine-Tuning (Optional):

- [Evaluation Metrics] → [Fine-Tune Model] → [Re-train Recommendation Model]

7. Deployment (Optional):

- [Fine-Tuned Model] → [Deploy Model] → [Real-Time Recommendations]

4. IMPLEMENTATION

4.1. CODE

```
# Library Supports Required
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import ast
from scipy import stats
from ast import literal_eval
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
import warnings; warnings.simplefilter('ignore')

# Loading the Dataset
apps = pd.read_csv('apps.tsv', delimiter='\t',encoding='utf-8')
user_history = pd.read_csv('user_history.tsv', delimiter='\t',encoding='utf-8')
jobs = pd.read_csv('jobs.tsv', delimiter='\t',encoding='utf-8', error_bad_lines=False)
users = pd.read_csv('users.tsv' ,delimiter='\t',encoding='utf-8')
test_users = pd.read_csv('test_users.tsv', delimiter='\t',encoding='utf-8')

# Exploring the DataSets
apps.head()
apps.columns
apps.shape
apps.info()

user_history.head()
user_history.columns
```

```
user_history.shape
user_history.info()

jobs.head(1)
jobs.columns
jobs.shape
jobs.info()

users.head(1)
users.columns
users.shape
users.info()

test_users.head()
test_users.columns
test_users.shape
test_users.info()
```

 # Splitting Data into Training and Testing based on the attribute. However being a recommendation model testing data can only be used id there is ground truth data corresponding each row in training data.

```
apps_training = apps.loc[apps['Split'] == 'Train']
apps_testing = apps.loc[apps['Split'] == 'Test']

user_history_training = user_history.loc[user_history['Split'] =='Train']
user_history_testing = user_history.loc[user_history['Split'] =='Test']

apps_training = apps.loc[apps['Split'] == 'Train']
apps_testing = apps.loc[apps['Split'] == 'Test']

users_training = users.loc[users['Split']=='Train']
users_testing = users.loc[users['Split']=='Test']
```

Data Visualizations

 # Plotting Country wise job Openings

jobs.groupby(['City','State','Country']).size().reset_index(name='Locationwise')

jobs.groupby(['Country']).size().reset_index(name='Locationwise').sort_values('Locationwise', ascending=False).head()

Country_wise_job=jobs.groupby(['Country']).size().reset_index(name='Locationwise').sort_values('Locationwise',ascending=False)

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

ax = sns.barplot(x="Country", y="Locationwise", data=Country_wise_job)

ax.set_xticklabels(ax.get_xticklabels(), rotation=90, ha="right")

ax.set_title('Country wise job openings')

plt.tight_layout()

plt.show()

 # Plotting State wise job Openings

jobs_US = jobs.loc[jobs['Country']=='US']

jobs_US[['City','State','Country']]

jobs_US.groupby(['City','State','Country']).size().reset_index(name='Locationwise').sort_values('Locationwise',ascending=False).head()

State_wise_job_US=jobs_US.groupby(['State']).size().reset_index(name='Locationwise').sort_values('Locationwise',ascending=False)

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

ax = sns.barplot(x="State", y="Locationwise",data=State_wise_job_US)

ax.set_xticklabels(ax.get_xticklabels(), rotation=90, ha="right")

ax.set_title('State wise job openings')

plt.tight_layout()

plt.show()

 # Plotting City wise job Openings

jobs_US.groupby(['City']).size().reset_index(name='Locationwise').sort_values('Locationwise',ascending=False)

City_wise_location=jobs_US.groupby(['City']).size().reset_index(name='Locationwise').sort_values('Locationwise',ascending=False)

City_wise_location_th = City_wise_location.loc[City_wise_location['Locationwise']>=12] plt.figure(figsize=(12,5))

ax = sns.barplot(x="City", y="Locationwise",data=City_wise_location_th.head(50))

ax.set_xticklabels(ax.get_xticklabels(), rotation=90, ha="right")

ax.set_title('City wise job openings')

plt.tight_layout()

plt.show()

 # Plotting State wise User Profiles

users_training.groupby(['Country']).size().reset_index(name='Locationwise').sort_values('Locationwise', ascending=False).head()

user_training_US = users_training.loc[users_training['Country']=='US']

user_training_US.groupby(['State']).size().reset_index(name='Locationwise_state').sort_values('Locationwise_state',ascending=False)

user_training_US_state_wise=user_training_US.groupby(['State']).size().reset_index(name='Locationwise_state').sort_values('Locationwise_state',ascending=False)

user_training_US_th=user_training_US_state_wise.loc[user_training_US_state_wise['Locationwise_state']>=12]

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

ax = sns.barplot(x="State", y="Locationwise_state",data=user_training_US_th.head(50))

ax.set_xticklabels(ax.get_xticklabels(), rotation=90, ha="right")

ax.set_title('State wise job seekers')

plt.tight_layout()

plt.show()

 # Plotting City wise User Profiles

user_training_US.groupby(['City']).size().reset_index(name='Locationwise_city').sort_values('Locationwise_city',ascending=False)

user_training_US_city_wise=user_training_US.groupby(['City']).size().reset_index(name='Locationwise_city').sort_values('Locationwise_city',ascending=False)

user_training_US_City_th=user_training_US_city_wise.loc[user_training_US_city_wise['Locationwise_city']>=12]

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

ax = sns.barplot(x="City", y="Locationwise_city",data=user_training_US_City_th.head(50))

ax.set_xticklabels(ax.get_xticklabels(), rotation=90, ha="right")

ax.set_title('State wise job seekers')

```
plt.tight_layout()
plt.show()
```

Finding Similar Job Profiles

```
#Data Imputation
jobs_US_base_line['Title'] = jobs_US_base_line['Title'].fillna('')
jobs_US_base_line['Description'] = jobs_US_base_line['Description'].fillna('')
jobs_US_base_line['Description'] = jobs_US_base_line['Title'] +
jobs_US_base_line['Description']

#Applying Algorithm
tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df=0.0001,
stop_words='english')
tfidf_matrix = tf.fit_transform(jobs_US_base_line['Description'])
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

# Getting Similar Job Profiles
jobs_US_base_line = jobs_US_base_line.reset_index()
titles = jobs_US_base_line['Title']
indices = pd.Series(jobs_US_base_line.index, index=jobs_US_base_line['Title'])
#indices.head(2)
def get_recommendations(title):
    idx = indices[title]
    #print (idx)
    sim_scores = list(enumerate(cosine_sim[idx]))
    #print (sim_scores)
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    job_indices = [i[0] for i in sim_scores]
    return titles.iloc[job_indices]

# Testing the Content Filtering Approach
get_recommendations('SAP Business Analyst / WM').head(10)
```

get_recommendations('Security Engineer/Technical Lead').head(10)

get_recommendations('Immediate Opening').head(10)

get_recommendations('EXPERIENCED ROOFERS').head(10)

Recommend jobs based on similar user profiles

Find out similar users -- Find out for which jobs they have applied -- suggest those job to the other users who shared similar user profile.
We are finding put similar user profile based on their degree type, majors and total years of experience.

- We will get to 10 similar users.
- We will find out which are the jobs for which these users have applied
- We take an union of these jobs and recommend the jobs users based

 # Feature Engineering

user_based_approach_US = users_training.loc[users_training['Country']=='US']

user_based_approach = user_based_approach_US.iloc[0:10000,:]

user_based_approach['DegreeType'] = user_based_approach['DegreeType'].fillna('')

user_based_approach['Major'] = user_based_approach['Major'].fillna('')

user_based_approach['TotalYearsExperience']=str(user_based_approach['TotalYearsExperienc e'].fillna(''))

user_based_approach['DegreeType']=user_based_approach['DegreeType']+user_based_appro ach['Major'] + user_based_approach['TotalYearsExperience']

 # Applying Algorithm

tf = TfidfVectorizer(analyzer='word',ngram_range=(1, 2),min_df=0.0001, stop_words='english')

tfidf_matrix = tf.fit_transform(user_based_approach['DegreeType'])

cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

 # Getting Similar User

user_based_approach = user_based_approach.reset_index()

userid = user_based_approach['UserID']

indices = pd.Series(user_based_approach.index, index=user_based_approach['UserID'])

def get_recommendations_userwise(userid):

```
    idx = indices[userid]
    #print (idx)
    sim_scores = list(enumerate(cosine_sim[idx]))
    #print (sim_scores)
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    user_indices = [i[0] for i in sim_scores]
    return user_indices[0:11]
```

 # The jobs for which these users have applied

```
def get_job_id(usrid_list):
    jobs_userwise = apps_training['UserID'].isin(usrid_list) #
    df1 = pd.DataFrame(data = apps_training[jobs_userwise], columns=['JobID'])
    joblist = df1['JobID'].tolist()
    Job_list = jobs['JobID'].isin(joblist)
    df_temp = pd.DataFrame(data = jobs[Job_list],
columns=['JobID','Title','Description','City','State'])
    return df_temp
```

 # Testing the User Based Approach :

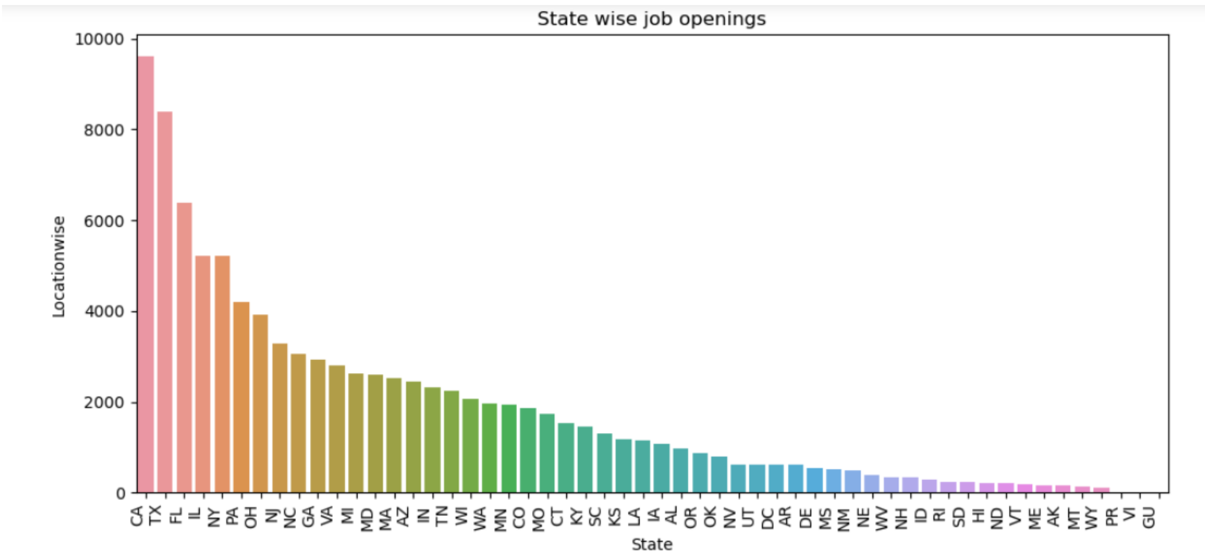
```
print ("-----Top 10 Similar users with userId: 47------")
get_recommendations_userwise(47)
get_job_id(get_recommendations_userwise(47))

print ("-----Top 10 Similar users with userId: 123------")
get_recommendations_userwise(123)
get_job_id(get_recommendations_userwise(123))
```

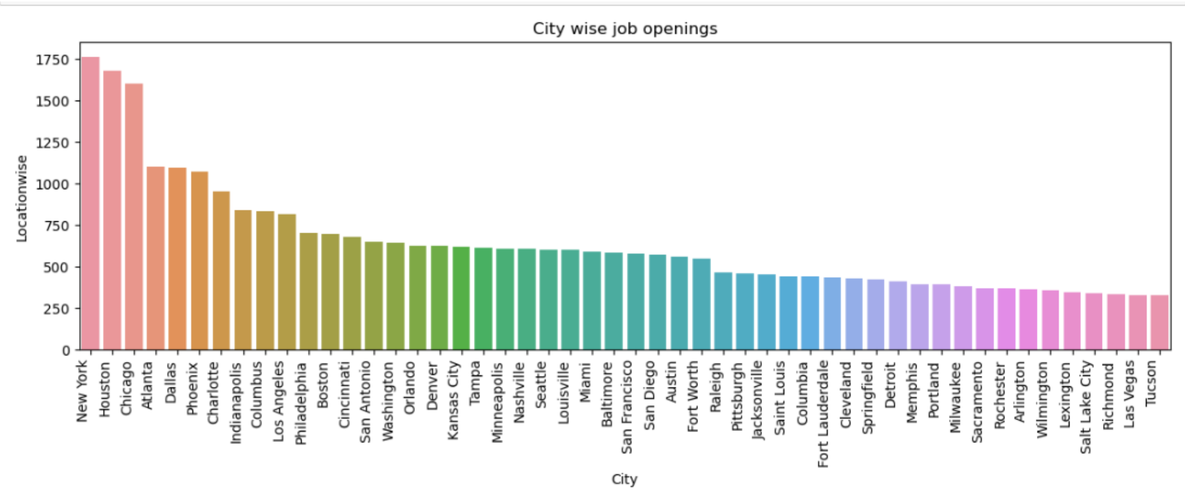
4.2. RESULT ANALYSIS

I. (EDA For Job Openings Feature)

A . Data Visualisations for Job Openings



I. A.1. Analysing the State Wise Job Openings

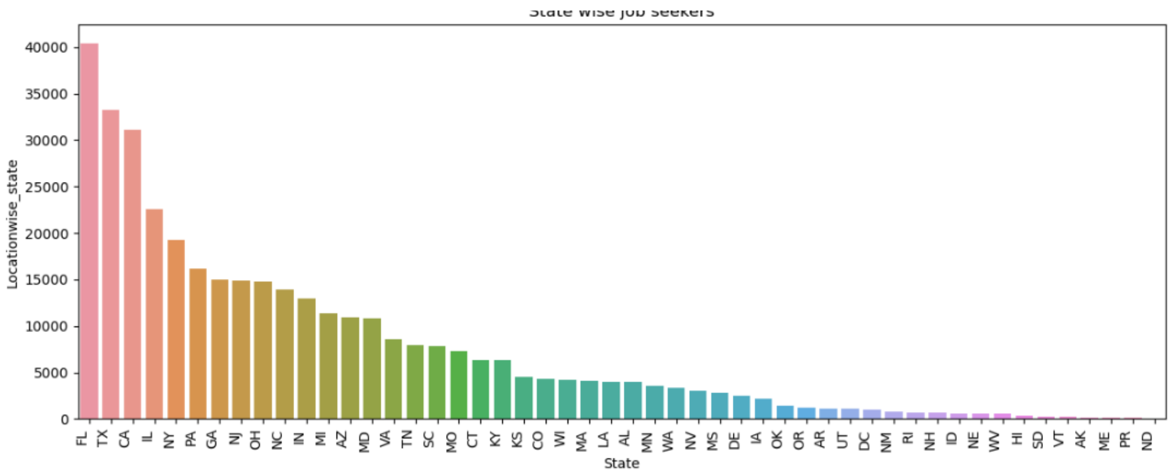


I.A.2. Analysing the City Wise Job Openings

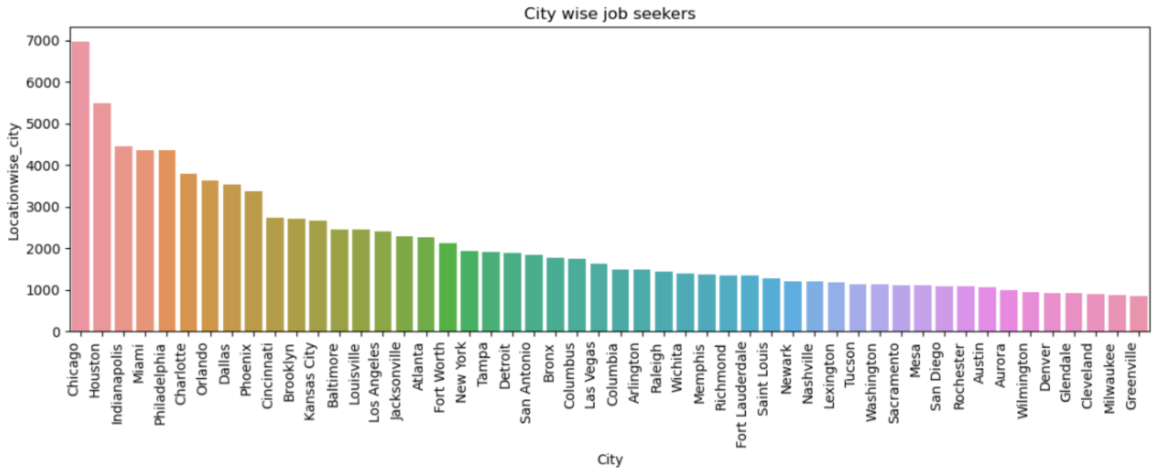
B. Observations:

- When we do analysis state-wise then **CA, TX, FL, IL and NY** are having more job opening than other states.
- When we do analysis city-wise then **Houston, New York, Chicago, Dallas, Atlanta and Phoenix** are having more jobs in comparison to the other cities.

II
(EDA for User profiles based on their location information)



II.A.1. Analysing the State Wise User Profiles



II.A.2. Analysing the State Wise User Profiles

B. Observations:

- When we do analysis state-wise then **FL, TX, CA, IL and NY** are having more user profiles than other states.
- When we do analysis city-wise then **Chicago, Houston, Indianapolis, Miami** are having more users profile in comparison to the other cities.

Result: The Cities with the most job openings are also the one with more Job seekers. Hence, we can integrate collaborative filtering along with content-based filtering.

III. RECOMMENDATION RESULTS :

A. Content Based Recommendation

```
In [175]: get_recommendations('Security Engineer/Technical Lead').head(10)

Out[175]: 0                        Security Engineer/Technical Lead
          79                                       AUTO TECHNICIAN
          126                                   Sales Professional
          182      LEASING AGENT-Luxury Rental Propery seeking ou...
          345                Tool Engineer - Injection Molding
          366          ASSISTANT BRANCH  MANAGER- NEW YORK, NY 10022
          393             Plastics Engineer - Injection Molding
          423          EDI Specialist - Gentran and .NET Developer
          434                                     Process Engineer
          490                Medical Biller 25 hrs incl Sun 9 - 3
          Name: Title, dtype: object
```

Fig. 1. Content Based Recommendation

for Security Engineer/Technical Lead

```
In [176]: get_recommendations('Immediate Opening').head(10)

Out[176]: 13                                     Immediate Opening
          971     Macy's South Coast Plaza, Costa Mesa, CA: Reta...
          1026    Macy's University Square, University Heights, ...
          1539      Graphic Designer for Postcard Marketing Company
          1606    Retail General Manager - Relocatable - $50K-$6...
          1683            Quality Manager - Sr. Quality Engineering
          1685                             Senior Medical Assistant
          1823                           Supervisor, OPI Department
          2058                        Vice President of Distribution
          2075                                   Licensed Life Agent
          Name: Title, dtype: object
```

Fig. 2. Content Based Recommendation

for Immediate Opening

B. Collaborative Filtering Recommendation

In [128…
```
print ("-----Top 10 Similar users with userId: 47------")
get_recommendations_userwise(47)
```

```
-----Top 10 Similar users with userId: 47------
```

Out[128…
```
[0, 79, 126, 182, 345, 366, 393, 423, 434, 490, 544]
```

In [167…
```
get_job_id(get_recommendations_userwise(47))
```

Out[167…

	JobID	Title	Description	City	State
905894	428902	Aircraft Servicer	Job Classification: Direct Hire \r\n\r...	Memphis	TN
975525	1098447	Automotive Service Advisor	<div>\r<div>Briggs Nissan in Lawrence Kansas h...	Lawrence	KS
980507	37309	Medical Lab Technician - High Volume Lab	<span>Position Title: <span> &...	Fort Myers	FL
986244	83507	Nurse Tech (CNA/STNA)	<p align="center"><b>Purpose of Your Job Posit...	Englewood	FL
987452	93883	Nurse Tech II (CNA/STNA)	<B>Nurse Tech II (CNA/STNA)</B> \r \rTh...	Fort Myers	FL
1000910	228284	REGISTERED NURSE – ICU	<p><strong><span><font face="">Registered Nurs...	Punta Gorda	FL
1007140	284840	Certified Nursing Assistant / CNA	<hr>\r<p style="text-align: center"><strong>Ce...	Saint Petersburg	FL
1007141	284841	Home Health Aide / HHA	<hr>\r<p style="text-align: center"><strong>Ho...	Saint Petersburg	FL

Fig 3. User based recommendation of job openings

5. CONCLUSION

In the culmination of the Enhanced Job Recommendation Model for Personalized and Collaborative Job Matching, a transformative solution was explored. This model redefines the job search landscape by seamlessly blending personalized and collaborative elements.

However, this study did not leverage the possibility of fine tuning after testing our recommending model. This is because to calculate evaluation metrics such as recall, precision and f1, we need ground_truth data that is the actual recommendations corresponding each entry in training dataset. Absence of such dataset makes the calculation if evaluation metrics out of scope of this project. Leveraging advanced machine learning techniques, this idea would excel at understanding individual user preferences, delivering tailored job suggestions based on unique skills and interests.

Beyond streamlining the job search process, the model creates a collaborative environment. Users not only receive personalized recommendations but also have the opportunity to connect with like-minded individuals, fostering a supportive community within the platform.

In conclusion, the Enhanced Job Recommendation Model sets a new standard in intelligent and user-centric job recommendation systems. Its fusion of personalization and collaboration promises to shape a more efficient and fulfilling future for job seekers globally.

6. REFERENCES

1. GeeksforGeeks: https://www.geeksforgeeks.org/
2. Analytics Vidhya: https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/
3. Medium Article : https://towardsdatascience.com/natural-language-processing-feature-engineering-using-tf-idf-e8b9d00e7e76
4. BlogPosts: https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/
5. Scikit-learn Documentation: https://scikit-learn.org/stable/documentation.html
6. Udemy Course: Machine Learning A-Z™: AI, Python https://www.udemy.com/course/machinelearning/