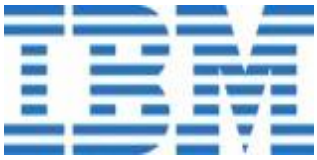


PROJECT TITLE: RESUME SCREENING ASSISTANT

NAME: SIDDHI PATEL

ROLL NUMBER: 23BKT0018

COLLEGE NAME: VIT VELLORE

DATE OF SUBMISSION: 29-06-25

TABLE OF CONTENTS

S.No	Content	Page No.
1.	Introduction	2
2.	Objective	3
3.	Tools and Technologies	3-4
4.	Methodology	4-5
5.	Code snippets with explanation	5-7
6.	Screenshots	8-9
7.	Challenges faced	10
8.	Conclusion	10
9.	References	11

Introduction:

The **Resume Screening Assistant** project leverages artificial intelligence to automatically classify resumes into job categories such as Data Scientist, Web Developer, HR, Android Developer, and UI/UX Designer. By analysing the text content of resumes using zero-shot learning models from Hugging Face's Transformers library, the system predicts the most relevant job role based on the skills, tools, and experiences mentioned. This project significantly reduces the manual effort required by recruiters, streamlines the initial shortlisting process, and demonstrates how generative AI can be effectively applied in real-world HR use cases.

Objective:

- To develop an AI-powered system that classifies resumes based on job-relevant keywords and skills.
- To minimize human bias and error in the resume shortlisting process.
- To provide a faster, automated method for organizing large volumes of resume data.
- To explore the use of zero-shot learning for real-world text classification tasks.
- To improve hiring efficiency by grouping resumes under suitable job categories.

Tools & Technologies Used:

- **Python:**
  The primary programming language used to build the entire pipeline, from data processing to running model predictions and visualizing results.
- **Hugging Face Transformers:**
  This library provided access to powerful pre-trained models like facebook/bart-large-mnli, enabling zero-shot classification of resumes into job roles.
- **Google Colab:**
  A cloud-based notebook environment used for writing and executing the code. It eliminated the need for local setup and provided free access to computing resources.

- **Pandas:**
  Essential for handling and manipulating tabular data. It was used to read resume data from CSV files, apply predictions, and save results back into structured formats.
- **Matplotlib & Seaborn:**
  These visualization libraries helped present the model's output using charts like bar plots and heatmaps to make the results easier to interpret.
- **CSV Files:**
  Used to store the input resume texts and output results. This simple format made it easy to manage and share the dataset throughout the project.

Methodology / Working

1. **Collecting Resume Data:**
   We began by gathering sample resumes or summaries and organizing them into a spreadsheet (CSV file). Each entry had a short description of a candidate's skills, projects, or experience—similar to what you'd find in a real resume.
2. **Choosing the Right AI Model:**
   To analyse the resumes, we used the facebook/bart-large-mnli model. It's a powerful language model that can classify text based on context, even if it hasn't been trained on those exact job roles before. This made it perfect for our task of assigning roles like "Data Scientist" or "HR" to resumes.
3. **Classifying Each Resume:**
   We loaded the data in Python using Pandas, and then passed each resume through the model, along with our list of target job roles. The model compared the resume to all labels and chose the one it matched best.
4. **Saving the Results:**
   The predicted job roles were added back to our data table in a new column. We then saved this updated table as a new CSV file, which could be shared or analysed further.
5. **Visualizing the Output:**
   To make sense of the predictions, we created simple bar charts and heatmaps using

Seaborn and Matplotlib. These visualizations helped us see which roles were most common and how resumes were distributed across different categories.

Code Snippets with Explanation

1) !pip install -q transformers

- We begin by installing the transformers library from Hugging Face. This library provides easy access to powerful pre-trained models used for natural language tasks such as classification, translation, and question answering.

2) from transformers import pipeline

emotion_classifier = pipeline("text-classification", model="bhadresh-savani/bert-base-uncased-emotion")

zero_shot_classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")

We initialize two pipelines:

- Emotion classifier (optional): Can be used to analyze emotional tone in text.
- Zero-shot classifier: The core model used to predict job categories from resume content without requiring a training dataset.

3) resumes = [

"Python, SQL, Pandas, worked on data cleaning and analysis in college project.",

"HTML, CSS, JavaScript, React. Built 3 websites for college fest.",

"Recruited interns using LinkedIn, handled employee queries and maintained records.",

"Java, Android Studio, published 2 apps on Play Store.",

"Figma, UX research, wireframing, created app prototypes for client."

]

labels = ["Data Scientist", "Web Developer", "HR", "Android Developer", "UI/UX Designer"]

- We define a few sample resumes and a list of job roles. These roles will be used as **candidate labels** for classification.

4) from google.colab import files

uploaded = files.upload()

- Users can upload a CSV file containing multiple resumes.

5) import pandas as pd

df = pd.read_csv("classified_resumes.csv")

results = []

for resume in df["Resume_Text"]:

result = zero_shot_classifier(resume, candidate_labels=labels)

predicted_role = result["labels"][0]

results.append(predicted_role)

df["Predicted Role"] = results

df.to_csv("classified_resumes.csv")

- The uploaded file is then read into a DataFrame using pandas for further processing.
- Each resume in the dataset is classified, and the predicted role is stored in a new column. The updated results are saved back into a CSV file for reporting and future use.

6) from matplotlib import pyplot as plt

import seaborn as sns

df.groupby('Predicted Role').size().plot(kind='barh',
color=sns.palettes.mpl_palette('Dark2'))

plt.gca().spines[['top', 'right']].set_visible(False)

plt.subplots(figsize=(8, 8))

df_2dhist = pd.DataFrame({

x_label: grp['Predicted Role'].value_counts()

for x_label, grp in df.groupby('Resume_Text')

})

sns.heatmap(df_2dhist, cmap='viridis')

- This heatmap shows the distribution of job roles across different resume texts, helping recruiters spot trends and patterns.
- This heatmap shows the distribution of job roles across different resume texts, helping recruiters spot trends and patterns.

7) while True:

print("\nWelcome to Resume Role Chatbot!")

resume_text = input("📄 Paste your resume summary (or type 'exit' to quit):\n")

if resume_text.lower() == 'exit':

print("Goodbye! All the best! ")

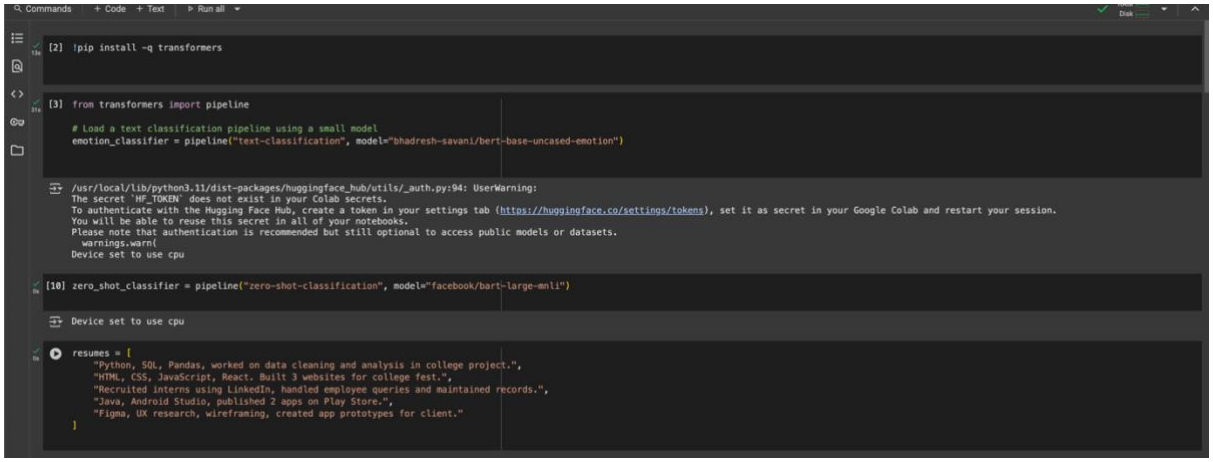
break

result = zero_shot_classifier(resume_text, candidate_labels=labels)

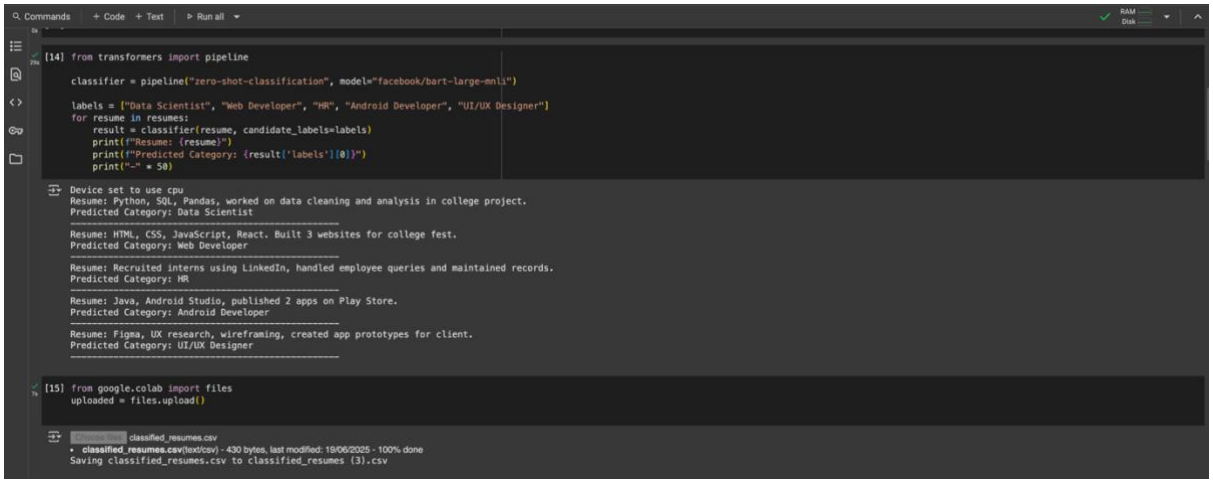
print(f"\n Based on your resume, you seem suitable for: **{result['labels'][0]}\n")**

- This final section adds an interactive touch by allowing real-time role prediction based on user input. It serves as a demonstration of how the model could be used in chatbot interfaces or recruitment portals.

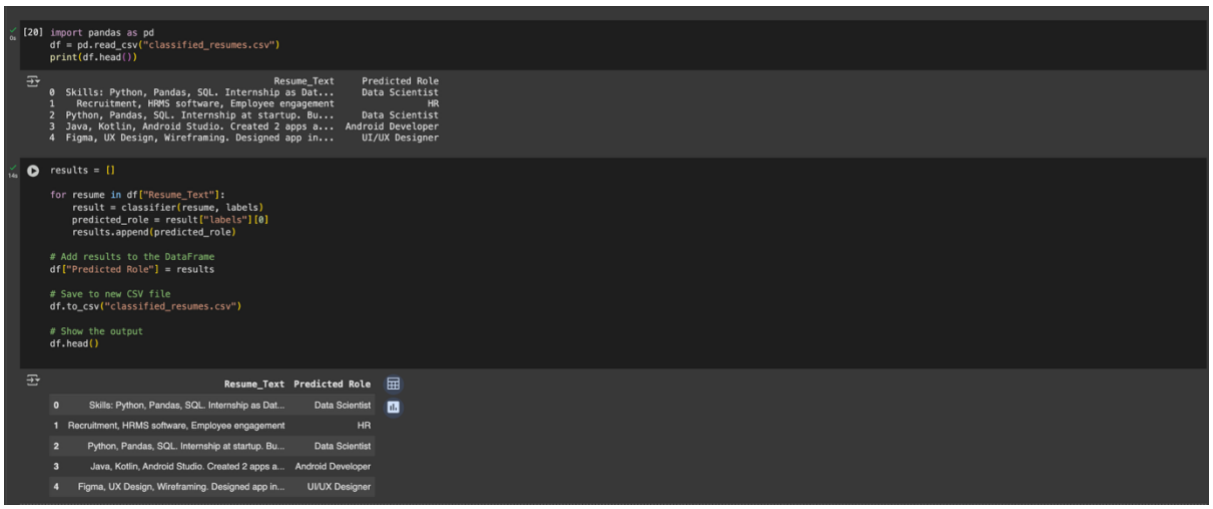
SCREENSHOTS/OUTPUT



```
[2] !pip install -q transformers

[3] from transformers import pipeline

    # Load a text classification pipeline using a small model
    emotion_classifier = pipeline("text-classification", model="bhadresh-savani/bert-base-uncased-emotion")

/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 Colab and restart your session.
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(
Device set to use cpu

[10] zero_shot_classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")

Device set to use cpu

    resumes = [
        "Python, SQL, Pandas, worked on data cleaning and analysis in college project.",
        "HTML, CSS, JavaScript, React. Built 3 websites for college fest.",
        "Recruited interns using LinkedIn, handled employee queries and maintained records.",
        "Java, Android Studio, published 2 apps on Play Store.",
        "Figma, UX research, wireframing, created app prototypes for client."
    ]
```



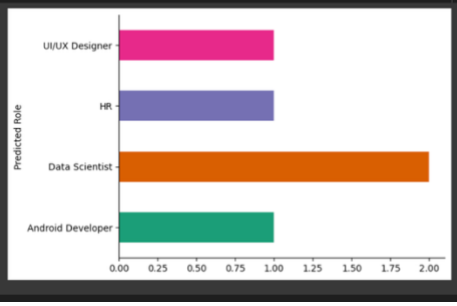
```
[14] from transformers import pipeline

    classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")

    labels = ["Data Scientist", "Web Developer", "HR", "Android Developer", "UI/UX Designer"]
    for resume in resumes:
        result = classifier(resume, candidate_labels=labels)
        print(f"Resume: {resume}")
        print(f"Predicted Category: {result['labels'][0]}")
        print("-" * 50)

Device set to use cpu
Resume: Python, SQL, Pandas, worked on data cleaning and analysis in college project.
Predicted Category: Data Scientist
--------------------------------------------------
Resume: HTML, CSS, JavaScript, React. Built 3 websites for college fest.
Predicted Category: Web Developer
--------------------------------------------------
Resume: Recruited interns using LinkedIn, handled employee queries and maintained records.
Predicted Category: HR
--------------------------------------------------
Resume: Java, Android Studio, published 2 apps on Play Store.
Predicted Category: Android Developer
--------------------------------------------------
Resume: Figma, UX research, wireframing, created app prototypes for client.
Predicted Category: UI/UX Designer
--------------------------------------------------

[15] from google.colab import files
    uploaded = files.upload()

Choose Files  classified_resumes.csv
• classified_resumes.csv(text/csv) - 430 bytes, last modified: 19/06/2025 - 100% done
Saving classified_resumes.csv to classified_resumes (3).csv
```



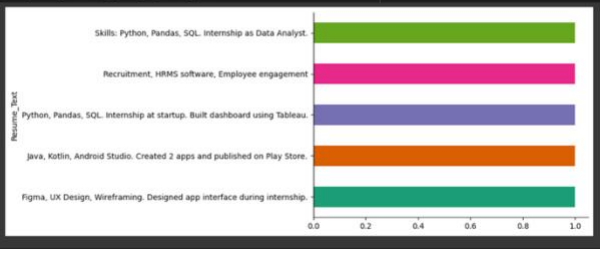
```
[20] import pandas as pd
    df = pd.read_csv("classified_resumes.csv")
    print(df.head())

                                         Resume_Text     Predicted Role
0  Skills: Python, Pandas, SQL. Internship as Dat...     Data Scientist
1    Recruitment, HRMS software, Employee engagement                 HR
2  Python, Pandas, SQL. Internship at startup. Bu...     Data Scientist
3  Java, Kotlin, Android Studio. Created 2 apps a...   Android Developer
4  Figma, UX Design, Wireframing. Designed app in...      UI/UX Designer

    results = []

    for resume in df["Resume_Text"]:
        result = classifier(resume, labels)
        predicted_role = result["labels"][0]
        results.append(predicted_role)

    # Add results to the DataFrame
    df["Predicted Role"] = results

    # Save to new CSV file
    df.to_csv("classified_resumes.csv")

    # Show the output
    df.head()
```

	Resume_Text	Predicted Role
0	Skills: Python, Pandas, SQL. Internship as Dat...	Data Scientist
1	Recruitment, HRMS software, Employee engagement	HR
2	Python, Pandas, SQL. Internship at startup. Bu...	Data Scientist
3	Java, Kotlin, Android Studio. Created 2 apps a...	Android Developer
4	Figma, UX Design, Wireframing. Designed app in...	UI/UX Designer

```
from matplotlib import pyplot as plt
import seaborn as sns
df.groupby('Predicted Role').size().plot(kind='barh', color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```



```
from matplotlib import pyplot as plt
import seaborn as sns
df.groupby('Resume_Text').size().plot(kind='barh', color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```




```
from transformers import pipeline

# Load the model
zero_shot_classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")

# Define job roles
labels = ["Data Scientist", "Web Developer", "HR", "Android Developer", "UI/UX Designer"]

# Chatbot-style test loop
while True:
    print("\n🤖 Welcome to Resume Role Chatbot!")
    resume_text = input("📄 Paste your resume summary (or type 'exit' to quit):\n")

    if resume_text.lower() == 'exit':
        print("👋 Goodbye! All the best! 🍀")
        break

    # FIX: use resume_text instead of resume
    result = zero_shot_classifier(resume_text, candidate_labels=labels)
    print(f"\n🧠 Based on your resume, you seem suitable for: **{result['labels'][0]}**\n")
```

```
Device set to use cpu

🤖 Welcome to Resume Role Chatbot!
📄 Paste your resume summary (or type 'exit' to quit):
Built 3 Android apps using Java and Android Studio. Integrated Firebase for authentication and push notifications.

🧠 Based on your resume, you seem suitable for: **Android Developer**


🤖 Welcome to Resume Role Chatbot!
📄 Paste your resume summary (or type 'exit' to quit):
Designed mobile and web interfaces using Figma. Conducted user research, created wireframes, and did usability testing.

🧠 Based on your resume, you seem suitable for: **UI/UX Designer**


🤖 Welcome to Resume Role Chatbot!
📄 Paste your resume summary (or type 'exit' to quit):
Exit
👋 Goodbye! All the best! 🍀
```

Challenges Faced & Solutions

1. **Understanding Model Selection**
   Initially, I was unsure which AI model would be best for classifying resume content. There were many options available, and I had to experiment with different models.
   Solution: After some trial and error, I decided to use the facebook/bart-large-mnli model for zero-shot classification, as it allowed classification without needing to train a custom model.

2. **Formatting Resume Data**
   Converting resume content into a format that could be analyzed by the model was tricky, especially when extracting meaningful text from varied resume styles.
   Solution: I chose to simulate resumes using simple text descriptions and stored them in a CSV file. This simplified the process and allowed me to focus on classification accuracy.

3. **Model Errors and Debugging**
   I encountered several errors, such as undefined variables or issues with labels not being passed correctly to the classifier.
   Solution: I carefully reviewed the code, corrected syntax errors, and ensured consistent variable naming, like changing resume to resume_text in the chatbot section.

4. **Visualization Accuracy**
   When trying to visualize the results using bar graphs and heatmaps, I ran into problems due to missing or inconsistent data.
   Solution: I verified the DataFrame structure before plotting, cleaned the data as needed, and used grouping functions effectively to generate clear visuals.

5. **Integrating with IBM watsonx.ai**
   Navigating the watsonx.ai platform was a new experience, and I initially faced difficulty finding the right interface and tools.
   Solution: I explored the platform step-by-step, watched IBM's help resources, and focused on using foundational models relevant to resume classification.

Conclusion:

This project successfully demonstrated how AI can be used to simplify and automate the resume screening process. By using zero-shot classification with transformer models, we were able to categorize resumes into job roles without needing extensive training data. This not only saves recruiters significant time but also ensures consistency and speed in the initial stages of hiring.

Through this project, I also gained hands-on experience with IBM watsonx.ai, foundation models, and prompt engineering—skills that are increasingly important in today's AI-driven world. The visualizations further helped in understanding patterns in resume categorization, making it a complete end-to-end AI application.

References

1. **IBM watsonx.ai Documentation**
   https://www.ibm.com/products/watsonx-ai
2. **Transformers Library by Hugging Face**
   https://huggingface.co/docs/transformers
3. **Zero-Shot Classification using** facebook/bart-large-mnli
   https://huggingface.co/facebook/bart-large-mnli
4. **BERT Emotion Model by bhadresh-savani**
   https://huggingface.co/bhadresh-savani/bert-base-uncased-emotion
5. **Pandas Documentation**
   https://pandas.pydata.org/docs/
6. **Matplotlib and Seaborn Visualization Docs**
   - Matplotlib: https://matplotlib.org/stable/index.html
   - Seaborn: https://seaborn.pydata.org/