

Configuration Manual

MSc Research Project
M.Sc. Data Analytics

Jareen Sayma Haque
Student ID: X23383593

School of Computing
National College of Ireland

Supervisor: Abid Yaqoob

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: Jareen Sayma Haque

 X23383593
Student ID:
Programme: MSc in Data Analytics **Year:** 2024-2025
 Research Practicum
Module:
 Abid Yaqoob
Lecturer:
Submission Due Date: 11th August 2025
Project Title: Using Deep Learning and Transformer Models to Identify
 Inconsistencies Between Interview Responses and Resume Claims

 14
Word Count: **Page Count:**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Jareen Sayma Haque

 11th August 2025
Date:

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	✓
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	✓
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	✓

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Jareen Sayma Haque
Student ID: X23383593

1. System Configuration

I.1 Hardware

Processor: Intel(R) Core (TM) i7-8550U CPU @ 1.80GHz (1.99 GHz)
Installed RAM 16.0 GB (15.9 GB usable)
System type 64-bit operating system, x64-based processor

I.2 Operating System

Edition Windows 11 Home
Version 24H2
OS build 26100.4652
Experience Windows Feature Experience Pack 1000.26100.128.0

2. Software Requirements

Software's required for build and execution:

- **Integrated Development Environment** : Jupyter Notebook (streamlit_env)
- **Scripting Language** : Python 3.12.3
- **Storage** : Local folder
- **Surveying tool** : Google Form
- **Synthetic data generation tool** : Gemini Flash 3
- **Other Tools** : Notepad ++, Overleaf, Excel

3. Libraries

Various libraries were installed and used in the environment to run the project.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import ast
from sentence_transformers import SentenceTransformer, util
from sentence_transformers.cross_encoder import CrossEncoder
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix, roc_curve, precision_recall_curve
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM, AutoModelForSequenceClassification
import torch
from tqdm import tqdm
from sentence_transformers import InputExample
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt
import seaborn as sns
import lime
from lime.lime_text import LimeTextExplainer
import shap
```

```
import os
os.environ["OMP_NUM_THREADS"] = "1"

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import ast # To parse stringified lists from the CSV
import openpyxl
from wordcloud import WordCloud
from sklearn.feature_extraction.text import CountVectorizer
from scipy.stats import ttest_ind
import textstat

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.manifold import TSNE
from collections import Counter
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
import nltk
nltk.download('averaged_perceptron_tagger_eng')
sns.set_style('whitegrid')
```

Figure 1 Libraries used

4. Folder Structure

This is the root folder structure below consisting of all the demo artifacts including code, dataset, report and configure manual.

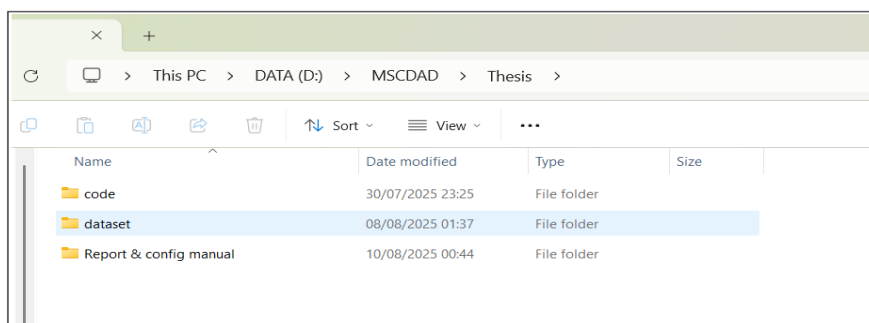


Figure 2 root folder

Inside the code folder, all the codes are structured in 3 subfolders for better clarity,

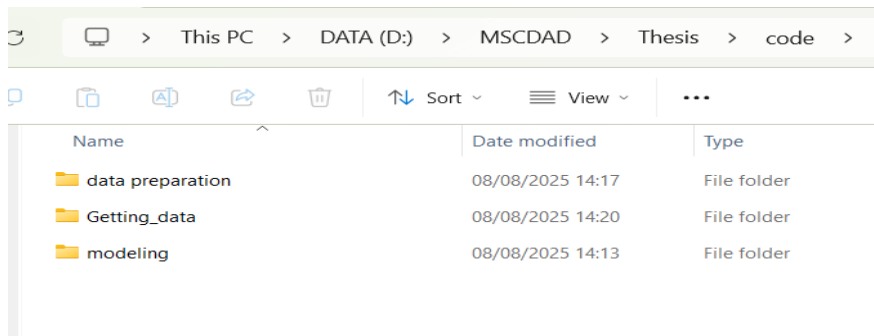


Figure 3 code folder

The dataset folder consists of all the dataset that was used and created throughout the whole procedure.

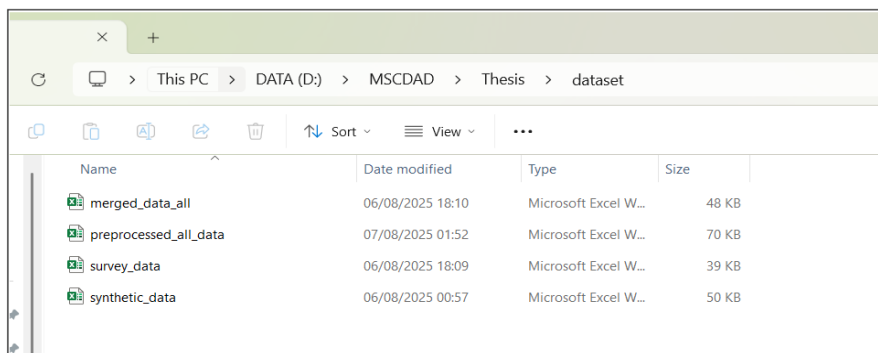


Figure 4 Dataset Folder

Sample of AI_prompt that was used in this data collection stage, is inside the folder below, code ---- getting_data --- “AI_tool_prompt_flash_gemini.docx”

Directory:

The root folder directory structure including all files are given below,

```

root/
├── datasets/
│   ├── survey_data.xlsx
│   ├── merged_data_all.xlsx
│   ├── preprocessed_all_data.xlsx
│   └── synthetic_data.xlsx
├── code/
│   ├── getting_data/
│   │   ├── AI_tool_prompt_flash_gemini.docx
│   │   ├── GeneratingSurvey_Data_Flash.ipynb
│   │   └── mergedfile_alldata.ipynb
│   ├── data_preparation/
│   │   ├── Preprocessing_merged_data_.ipynb
│   │   ├── rawEDA.ipynb
│   │   └── EDA_preprocessed.ipynb
│   └── modeling/
│       └── modeling.ipynb

```

5. Workflow to replicate the experiment

I. Setting Up Your Environment

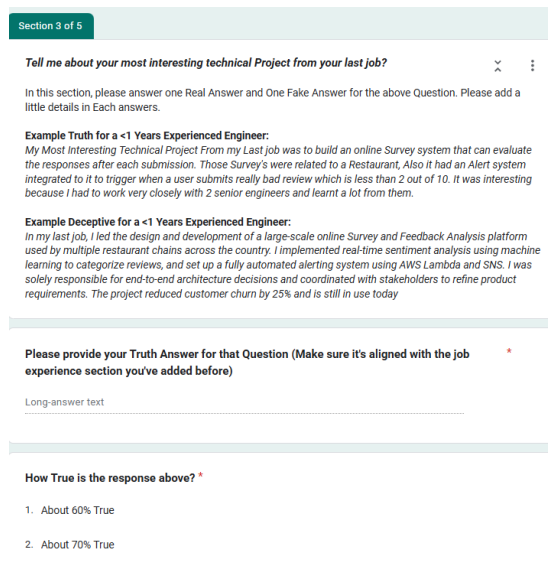
- Ensure that all the required software and Python libraries are installed.
- Open Jupyter Notebook.
- Set the working directory to the folder containing the dataset files.

II. Data Collection and Augmentation

Primary dataset is collected through google Survey form-

Input file: Survey Form Link,

<https://docs.google.com/forms/d/e/1FAIpQLSehZZSICX7eH3elanDJvcUtPqNMpu5XMp33kZ2bdPhg9zSuHA/viewform?usp=sharing&oid=109136116658387132990>



Section 3 of 5

Tell me about your most interesting technical Project from your last job?

In this section, please answer one Real Answer and One Fake Answer for the above Question. Please add a little details in Each answers.

Example Truth for a <1 Years Experienced Engineer:
My Most interesting Technical Project From my Last Job was to build an online Survey system that can evaluate the responses after each submission. Those Survey's were related to a Restaurant, Also it had an Alert system integrated to it to trigger when a user submits really bad review which is less than 2 out of 10. It was interesting because I had to work very closely with 2 senior engineers and learnt a lot from them.

Example Deceptive for a <1 Years Experienced Engineer:
In my last job, I led the design and development of a large-scale online Survey and Feedback Analysis platform used by multiple restaurant chains across the country. I implemented real-time sentiment analysis using machine learning to categorize reviews, and set up a fully automated alerting system using AWS Lambda and SNS. I was solely responsible for end-to-end architecture decisions and coordinated with stakeholders to refine product requirements. The project reduced customer churn by 25% and is still in use today

Please provide your Truth Answer for that Question (Make sure it's aligned with the job experience section you've added before)

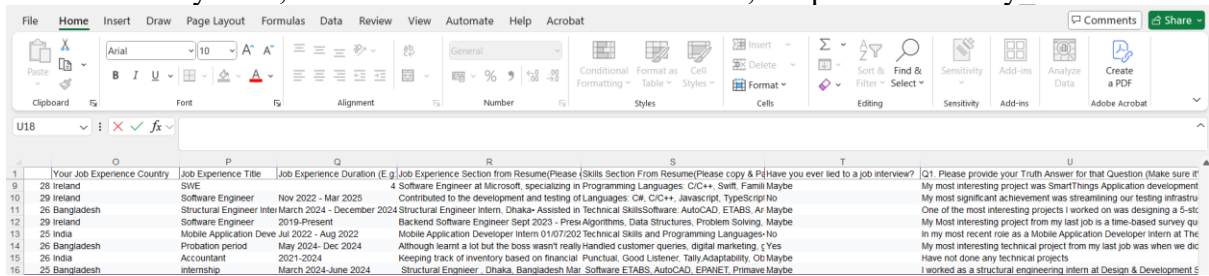
Long-answer text

How True is the response above? *

1. About 60% True
2. About 70% True

Figure 5 Google Survey Form

from the survey form, dataset is created as an excel file as, Output file: "survey_data.xlsx"



	O	P	Q	R	S	T	U
1	Your Job Experience Country	Job Experience Title	Job Experience Duration (E.g.	Job Experience Section from Resume(Please	Skills Section From Resume(Please copy & P	Have you ever led to a job interview?	Q1. Please provide your Truth Answer for that Question (Make sure it
9	Ireland	SWE	Nov 2022 - Mar 2025	4 Software Engineer at Microsoft, specializing in Programming Languages: C/C++, Swift, Fami	Maybe		My most interesting project was SmartTrangs Application Development
10	Ireland	Software Engineer	Nov 2022 - Mar 2025	Contributed to the development and testing of Languages: C#, C/C++, Javascript, TypeScript No			My most significant achievement was streamlining our testing infrastru
11	Bangladesh	Structural Engineer Intern	March 2024 - December 2024	Structural Engineer Intern, Dhaka- Assisted in Technical Skills:Software: AutoCAD, ETABS, Ar	Maybe		One of the most interesting projects I worked on was designing a 5-stc
12	Ireland	Software Engineer	2019-Present	Backend Software Engineer Sept 2023 - Presn Algorithms, Data Structures, Problem Solving, Maybe			My Most interesting project from my last job is a time-based survey qu
13	India	Mobile Application Deve	Jul 2022 - Aug 2022	Mobile Application Developer Intern 01/07/2022 Technical Skills and Programming Languages-No			in my most recent role as a Mobile Application Developer intern at The
14	Bangladesh	Probation period	May 2024 - Dec 2024	Although learnt a lot but the boss wasn't really Handled customer queries, digital marketing, Yes			My most interesting technical project from my last job was when we dic
15	India	Accountant	2021-2024	Keeping track of inventory based on financial Punctual, Good Listener, Tally, Adaptability, Ob	Maybe		Have not done any technical projects
16	Bangladesh	internship	March 2024-June 2024	Structural Engineer - Dhaka, Bangladesh Mar. Software ETABS, AutoCAD, EPANET, Primavera	Maybe		I worked as a structural engineering intern at Design & Development S

Figure 6 Survey dataset

Gemini flash 2.5 was used as an AI tool to generate synthetic data based on the human survey data excel file, a sample of prompt is below,

Data is merged programmatically into excel file. Files used in this stage are below,

- Input file: "code\Getting_data\mergedfile_alldata.ipynb"
- Output file: "dataset\merged_data_all.xlsx"

```

jupyter mergedfile_alldata Last Checkpoint: 4 days ago
File Edit View Run Kernel Settings Help
Python 3 (ipykernel)

[1]: Import pandas as pd

# Load the files
file1_path = "D:\MSCDMD\Thesis\dataset\survey_data.xlsx"
file2_path = "D:\MSCDMD\Thesis\dataset\synthetic_data.xlsx"

file1 = pd.read_excel(file1_path)
file2 = pd.read_excel(file2_path)

# Define the mapping (Standard Name: keywords to look for in column names)
column_map = {
    "Timestamp": ["timestamp"],
    "Consent": ["consent"],
    "Data_Usage": ["data", "train"],
    "Participation": ["freely agree", "pressured", "coerced"],
    "Study_Goals": ["goals", "purpose"],
    "Voluntary": ["entirely voluntary"],
    "Withdrawal": ["withdraw", "any time"],
    "Risks_Discomforts": ["risks", "discomforts"],
    "Beneficial": ["no direct or indirect financial"],
    "No_Harm": ["no physical", "psychological", "emotional harm"],
    "Vulnerable_Individuals": ["vulnerable", "minors", "reduced decision-making"],
    "Conflict_of_Interest": ["conflict of interest"],
    "Deception_Debrief": ["deception", "debriefed"],
    "Fun_Name": ["fun name"],
    "Age": ["age"],
    "Experience_Country": ["job experience country"],
    "Experience_Title": ["job experience title"],
    "Experience_Duration": ["job experience duration"],
    "Experience_Section": ["job experience section"],
    "Skill_Section": ["skill section"],
    "Have_You_Lead": ["Have you ever led"],
    "Q1_Truth_Answer": ["Truth Answer", "Q1"],
    "Q1_Truth_Answer_Truthfulness": ["How True", "Q1"]
}

```

Figure 10 Merging dataset code snippet

Merged dataset,

Timestamp	Data_Usage	Participation	Study_Goals	Voluntary	Withdrawal	Risks_Discomforts	Beneficial	No_Harm	Vulnerable_Individuals	Conflict_of_Interest	Deception_Debrief	Fun_Name	Age	Experience_Country	Experience_Title	Experience_Duration	Experience_Section	Skill_Section	Have_You_Lead	Q1_Truth_Answer	Q1_Truth_Answer_Truthfulness					
#####	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Agree	happy_joe	24	Malaysia	R&D Elective	2022	Completed	Program	No	One of the About 100	At my last AB					
#####	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Agree	L_vish_l_v	25	India	Data Engg	Sept 2022	Managed	Program	Maybe	My most is About 100	I worked at AB					
#####	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Agree	jesu adami	29	Bangladesh	Administrative	Feb 2020	Generational	Data	and Yes	One of the About 90%	In my last AB					
#####	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Agree	Milo	25	Ireland	Student	Sept 2022	-	OS/2022	-	Java C#	No	The most is About 90%	The most is AB			
#####	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Agree	moonhalf	23	Bangladesh	News Rep	August 2020	Correspondent	Computer	No	During my About 90%	During my AB					
#####	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Agree	confusion	27	Ireland	IT support	September	Production	Excel	Yes	My most is About 80%	My last AB					
#####	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Agree	power_pu	29	Ireland	Graduate	September	Generational	Data	and Maybe	One of the About 100	One of the AB					
#####	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Agree	funny_mn	28	Ireland	SWE	4	Software	Program	Maybe	My most is About 80%	I had a tea AB					
#####	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Agree	ledDragon	29	Ireland	Software	1 Nov 2022	Contribute	Language	No	My most is About 100	I had a High AB					
#####	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Agree	Mango ml	26	Bangladesh	Structural	March 2022	Structural	Technical	1	Maybe	One of the About 90%	In my last AB				
#####	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Agree	always_3n	29	Ireland	Software	1 2019	Pres	Backend	5	Algorithm	Maybe	My most is About 100	My most is AB			
#####	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Agree	Joni_nean	25	India	Mobile App	Jul 2022	-	Mobile App	Technical	No	In my most About 100	In my last AB				
#####	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Agree	Dnagmal_f	26	Bangladesh	Probation	May 2024	Although I	Handled	ci	Yes	My most is About 80%	In my last AB				
#####	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Agree	Nightmare	26	India	Accountant	2021	2024	Keeping	in	Punctual	1	Maybe	Have not is about 80%	Have not is AB		
2025-08-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	user_p004	32	UK	Data Analyst	2021	Pres	Data Analyst	SQL	Python	Yes	At Compas	100%	At Compas	50	
2025-08-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	user_p005	26	India	Product M	2020	Pres	Product M	Product M	Yes	At Compas	100%	At Compas	50		
2025-08-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	user_p006	29	UK	QA Engineer	2022	Pres	QA Engineer	Manual	Te	Yes	At Compas	100%	At Compas	50	
2025-08-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	user_p007	27	Spain	Software	1 2021	2024	Software	1	Go	Ruby	Yes	At Compas	100%	At Compas	50
2025-08-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	user_p008	30	Spain	Data Science	2022	2024	Data Science	Python	Yes	Yes	At Compas	100%	At Compas	50	
2025-08-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	user_p009	25	Pakistan	Frontend	1 2024	Pres	Frontend	React	Type	Yes	At Compas	100%	At Compas	50	
2025-08-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	user_p010	26	Spain	DevOps	En	2022	Pres	DevOps	En	AWS	Yes	At Compas	100%	At Compas	50
2025-08-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	user_p011	28	Nepal	Mobile De	2021	2023	Mobile De	Swift	Swift	Yes	At Compas	100%	At Compas	50	
2025-08-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	user_p012	32	Spain	Cloud Arch	2020	Pres	Cloud Arch	AWS	Azure	Yes	At Compas	100%	At Compas	50	

Figure 11 Merged Dataset

III. Pre-Processing & EDA

Merged dataset was pre-processed to make it ready for model training. Also 2 step EDA was done on the dataset, before and after preprocessing, to derive better and clearer insights from the data.

Data is pre-processed and stored in an excel file. Files used are below,

- Input file: "code\data preparation\Preprocessing_merged_data_.ipynb"

```
def preprocess_text_pipeline(text_series, include_stopwords=False):
    """
    Applies the full preprocessing pipeline to a pandas Series of text.

    Args:
        text_series (pd.Series): The pandas Series containing text data.
        include_stopwords (bool): If True, stop words will NOT be removed.
            Useful for tasks where context is critical.

    Returns:
        pd.Series: A Series of preprocessed text (list of lemmatized tokens).
    """
    print(f"Starting preprocessing for series: {text_series.name if hasattr(text_series, 'name') else 'Unnamed Series'}")
    # Step 1: Handle Missing Values
    print("- Handling missing values...")
    processed_series = handle_missing_values(text_series)

    # Step 2: Normalize Text (lowercase, remove punctuation/special chars)
    print("- Normalizing text (lowercasing, removing special characters)...")
    processed_series = processed_series.apply(normalize_text)

    # Step 3: Tokenization
    print("- Tokenizing text...")
    processed_series = processed_series.apply(tokenize_text)

    # Step 4: Stop Word Removal (conditional)
    if not include_stopwords:
        print("- Removing stop words...")
        processed_series = processed_series.apply(remove_stopwords)
    else:
        print("- Skipping stop word removal (as requested)...")

    # Step 5: Lemmatization
    print("- Lemmatizing tokens...")
    processed_series = processed_series.apply(lemmatize_tokens)

    print(f"Finished preprocessing for series: {text_series.name if hasattr(text_series, 'name') else 'Unnamed Series'}\n")
    return processed_series
```

Figure 12 Data Pre-processing

Output file: "dataset\preprocessed_all_data.xlsx"

Figure 13 Preprocessed data excel file

EDA:

Exploratory data analysis is performed through these two code files,

EDA on dataset **Before** Pre-processed:

- Input file:
 - I. "dataset \interview_data-.xlsx"
 - II. "dataset\Interview Response Survey (Responses).xlsx"
- Process: "code\data preparation\rawEDA.ipynb"

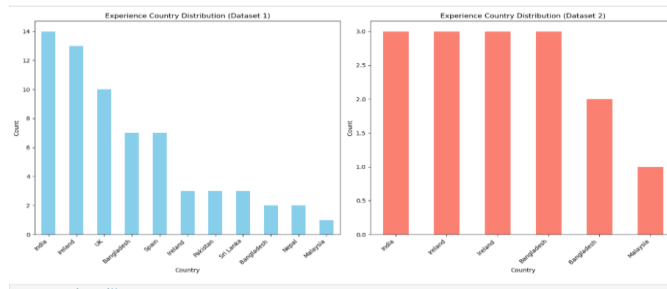


Figure 14 sample EDA snippet on raw datasets

EDA on dataset **After** Pre-processed:

- Input file: " dataset\preprocessed_all_data.xlsx"
- Process: "code\data preparation\EDA_preprocessed.ipynb"

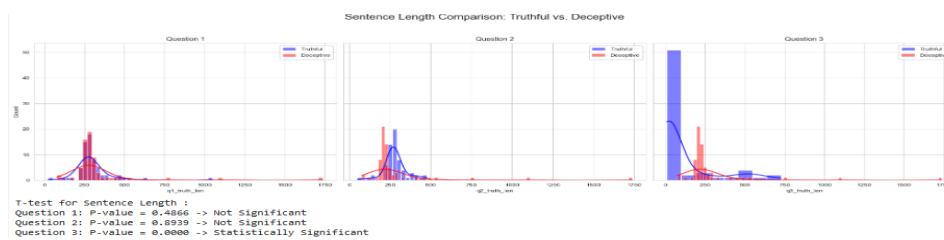


Figure 15 Sample EDA on pre-processed data

IV. Feature Engineering and Data Splitting

Pre-processed data set is loaded to feature engineering pipeline,

```

Jupyter modeling Last Checkpoint: 2 days ago
File Edit View Run Kernel Settings Help Trusted
JupyterLab Python 3 (ipykernel)

[6]: # Loading preprocessed dataset
try:
df = pd.read_excel('D:\MSCDAD\Thesis\dataset\preprocessed_all_data.xlsx')
except FileNotFoundError:
print("Error: 'preprocessed_data_new.xlsx - Sheet1.csv' not found.")
print("Please ensure the preprocessed data file is in the correct directory.")
exit()

# Feature Engineering: Reconstruct text from token Lists
# converting the string representation of Lists back into clean sentences so the models' internal tokenizers can process them correctly.
text_columns = ['processed_Q1_Truth_Answer', 'processed_Q1_Deceptive_Answer', 'processed_Q2_Truth_Answer',
'processed_Q2_Deceptive_Answer', 'processed_Q3_Truth_Answer', 'processed_Q3_Deceptive_Answer',
'processed_Experience_Section', 'processed_Skills_Section']

for col in text_columns:
if col in df.columns:
df[col.replace('processed_', 'clean_')] = df[col].apply(lambda x: ' '.join(ast.literal_eval(x)))
else:
print(f"Warning: Column '{col}' not found in the dataset.")

# Merging Experience and Skill section directly to a profile section
df['clean_Profile_Section'] = df['clean_Experience_Section'].fillna('') + ' ' + df['clean_Skills_Section'].fillna('')
df['clean_Profile_Section'] = df['clean_Profile_Section'].str.strip()

# Data Pairing
# This is the core restructuring step.
text_pairs = []
labels = []
sensitive_attrs = []

# Loop through all three question sets (Q1, Q2, Q3)
for index, row in df.iterrows():
for i in range(1, 4):
truth_col = f'clean_Q{i}_Truth_Answer'
deceptive_col = f'clean_Q{i}_Deceptive_Answer'
text_pairs.append([row[truth_col], row[deceptive_col]])
labels.append(1) # 1 represents "Consistent"
sensitive_attrs.append({'Age': row['Age'], 'Experience_Country': row['Experience_Country'], 'total_experience_years': row['total_experience_years']})

# Create an inconsistent pair from the deceptive answer
text_pairs.append([row[truth_col], row[deceptive_col]])
labels.append(0) # 0 represents "Inconsistent"

```

Figure 16 Feature engineering snippet

Then dataset is splitted through stratify to 80-20 ratio to prepare it for model training stage,

```

# data split into training and testing sets
X_train, X_test, y_train, y_test, sensitive_train, sensitive_test = train_test_split(
    full_df[['resume_segment', 'interview_answer']],
    full_df['label'],
    full_df[['Age', 'Experience_Country', 'total_experience_years']],
    test_size=0.2,
    random_state=42,
    stratify=full_df['label']
)

print(f"Total pairs created for modeling: {len(full_df)}")
print(f"Training set size: {len(X_train)}")
print(f"Test set size: {len(X_test)}")
print("\nSample Data Point:")
print(X_train.iloc[0])
print(f"Label: {y_train.iloc[0]}")
print("\nAnother Sample Data Point:")
print(X_train.iloc[1])
print(f"Label: {y_train.iloc[1]}")

y_true = np.asarray(y_test).ravel().astype(int)
print("Test positives:", y_true.sum(), "negatives:", (1-y_true).sum())

```

Total pairs created for modeling: 390
Training set size: 312
Test set size: 78

Figure 17 Dataset splitting

V. Model Training

The splitted data went through 3 architecture frameworks through rigorous model fine tuning,

Architectural Frameworks	Models Tested	Libraries
Bi-Encoder (SBERT)	all-MiniLM-L-6-v2 stsb-roberta-base	Hugging Face – <i>Sentence-Transformers</i>
Cross-Encoder (MiniLM)	cross-encoder/ms-marco-MiniLM-L-6-v2	Hugging Face – <i>Transformers</i> + <i>Sentence-Transformers</i>
Natural Language Inference (T5)	google/flan-t5-small	Hugging Face – <i>Transformers</i>

Final model, The pre-trained cross-encoder/ms-marco-MiniLM-L-6-v2, was loaded and fine-tuned on the training data. The process was configured to run for 4 epochs with a batch size of 4, warm-up period of 10 steps was used to stabilize the learning rate, and the maximum input sequence length was 384 tokens.

```

# Cross-Encoder using ms-marco-MiniLM-L-6-v2 model
# Setting max_length=384 to reduce the load to PC, our texts are aligned with this max token length

# Prepare training examples
train_samples = [InputExample(texts=[row.resume_segment, row.interview_answer], label=float(row.label))
                 for row in X_train.join(y_train).itertuples(index=False)]

cross_encoder_model = CrossEncoder("cross-encoder/ms-marco-MiniLM-L-6-v2", num_labels=1, max_length=384)

# DataLoader
train_data_loader = DataLoader(train_samples, shuffle=False, batch_size=4)

# Train
cross_encoder_model.fit(train_data_loader=train_data_loader,
                        epochs=4,
                        warmup_steps=10)

preds = cross_encoder_model.predict(list(zip(X_test['resume_segment'], X_test['interview_answer'])))

# Convert to class labels (threshold)
pred_labels = (np.array(preds) >= 0.5).astype(int)

# Evaluate
print("🔍 Cross-Encoder Evaluation Results")
print("Accuracy:", accuracy_score(y_test, pred_labels))
print("Precision:", precision_score(y_test, pred_labels))
print("Recall:", recall_score(y_test, pred_labels))
print("F1 Score:", f1_score(y_test, pred_labels))
print("ROC-AUC:", roc_auc_score(y_test, preds))
print("Confusion Matrix:\n", confusion_matrix(y_test, pred_labels))

```

Figure 18 Cross encoder training snippet

Other models like SBert & t5 went through threshold tuning to test accuracy,

```

Threshold = 0.3
Accuracy: 0.5256410256410257
Precision: 0.5166666666666667
Recall: 0.7948717948717948
F1 Score: 0.6262626262626263

Threshold = 0.4
Accuracy: 0.5384615384615384
Precision: 0.5348837209302325
Recall: 0.5897435897435898
F1 Score: 0.5609756097560976

Threshold = 0.5
Accuracy: 0.5641025641025641
Precision: 0.6086956521739131
Recall: 0.358974358974359
F1 Score: 0.45161290322580644

Threshold = 0.6
Accuracy: 0.5256410256410257
Precision: 0.6
Recall: 0.15384615384615385
F1 Score: 0.24489795918367346

```

Figure 19 SBert Threshold Tuning

VI. Evaluation

Each model evaluations metrics were tested where cross encoder performed best.

Cross-Encoder	Bi-Encoder	T5 NLI
Accuracy: 0.7307692307692307 Precision: 0.7368421052631579 Recall: 0.717948717948718 F1 Score: 0.7272727272727273 ROC-AUC: 0.836620644312952 Confusion Matrix: [[29 10] [11 28]]	<ol style="list-style-type: none"> All-MiniLM-L-6-v2 Accuracy: 0.5 Precision: 0.0 Recall: 0.0 F1 Score: 0.0 ROC-AUC: 0.5 Confusion Matrix: [[39 0] [39 0]] Stsb-roberta-base Accuracy: 0.5256410256410257 Precision: 0.625 Recall: 0.1282051282051282 F1 Score: 0.2127659574468085 ROC-AUC: 0.5256410256410257 Confusion Matrix: [[36 3] [34 5]] 	Accuracy: 0.5897435897435898 Precision: 1.0 Recall: 0.1794871794871795 F1 Score: 0.30434782608695654 ROC-AUC: 0.5897435897435898 Confusion Matrix: [[39 0] [32 7]]

Cross-encoders model evaluation is also showed through dashboard using matplotlib and seaborn,

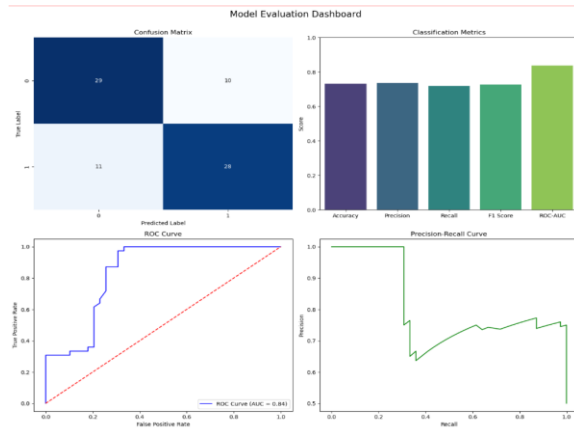


Figure 20 Cross-Encoder Model Evaluation Dashboard

VII. Fairness & Bias

Fairness audits were also performed, and SHAP/Lime Libraries were installed and used to show bias detection and mitigation through 2 examples from the result.

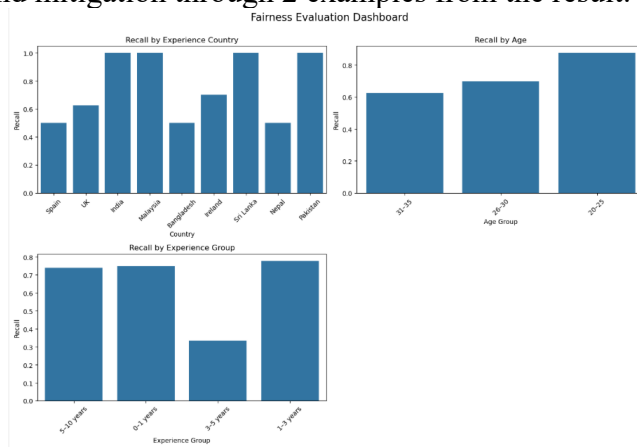


Figure 21 Fairness dashboard

```

lime_model_name = "cross-encoder/ms-marco-MiniLM-L-6-v2"
tokenizer = AutoTokenizer.from_pretrained(lime_model_name)
lime_model = AutoModelForSequenceClassification.from_pretrained(lime_model_name)

def predict_proba(text_pairs, batch_size=16):
    all_probs = []
    for i in range(0, len(text_pairs), batch_size):
        batch_pairs = text_pairs[i:i+batch_size]
        encodings = tokenizer.batch_encode_plus(
            batch_pairs, padding=True, truncation=True, return_tensors="pt"
        )
        with torch.no_grad():
            outputs = lime_model(**encodings)
            logits = outputs.logits # Shape: (batch_size, 1)
            scores = torch.sigmoid(logits).cpu().numpy() # Convert to [0, 1] range

            # Convert to 2-class probability: [No Match prob, Match prob]
            probs = np.hstack([1 - scores, scores]) # Shape: (batch_size, 2)
            all_probs.extend(probs)
    return np.array(all_probs)

import lime
import numpy as np
from lime.lime_text import LimeTextExplainer
class_names = ["No Match", "Match"]
explainer = LimeTextExplainer(class_names=class_names)
sample_data = full_df[['resume_segment', 'interview_answer']].head(2)
print(sample_data)

for idx, row in sample_data.iterrows():
    resume_text = row['resume_segment']
    interview_text = row['interview_answer']

```

Figure 22 Lime snippet

```

import shap
# Pre-computation Step
shap.initjs()
sample_data_short = full_df[['resume_segment', 'interview_answer']].head(2)
plt.figure(figsize=(20, 20)) # Width=20 inches, Height=20 inches (adjust as needed)

# Loop through your data
# Using 'break' at the end to demonstrate with just one example
for idx, row in sample_data_short.iterrows():
    resume_text = row['resume_segment']
    interview_text = row['interview_answer']

    # 1. GET PREDICTION DATA FIRST
    # Run a prediction on the text to get the probabilities and predicted class
    probabilities = predict_proba((resume_text, interview_text))[0]
    predicted_class_index = np.argmax(probabilities)

    predicted_class_name = class_names[predicted_class_index]
    confidence_score = probabilities[predicted_class_index]

    # 2. PRINT PREDICTION CONTEXT
    print("---- Prediction Details ----")
    print('---- Original Resume Experience ----')
    print(df[['Experience_Section']][0])
    print('---- Original Resume Skills ----')
    print(df[['Skills_Section']][0])

    print(f"---- CLEANED TEXT ANALYZED: --- '{interview_text}'")
    print(f"---- PREDICTED CLASS: --- '{predicted_class_name}' (Class Index: {predicted_class_index})")
    print(f"---- CONFIDENCE SCORE: --- {confidence_score:.4f}")
    print("==== So = \"\n\"")

    # 3. RUN SHAP EXPLAINER
    def shap_predict(texts):
        text_pairs = [(resume_text, text) for text in texts]
        return predict_proba(text_pairs)

    explainer = shap.Explainer(shap_predict, tokenizer)
    shap_values = explainer(interview_text)

```

Figure 23 SHAP code snippet

Through the Lime and SHAP framework, Bias is explained,

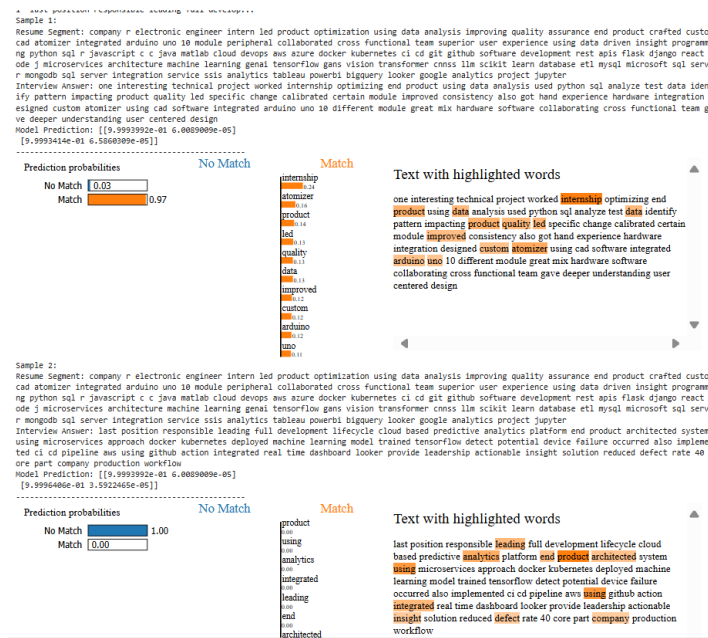


Figure 24 LIME Explainability Generated

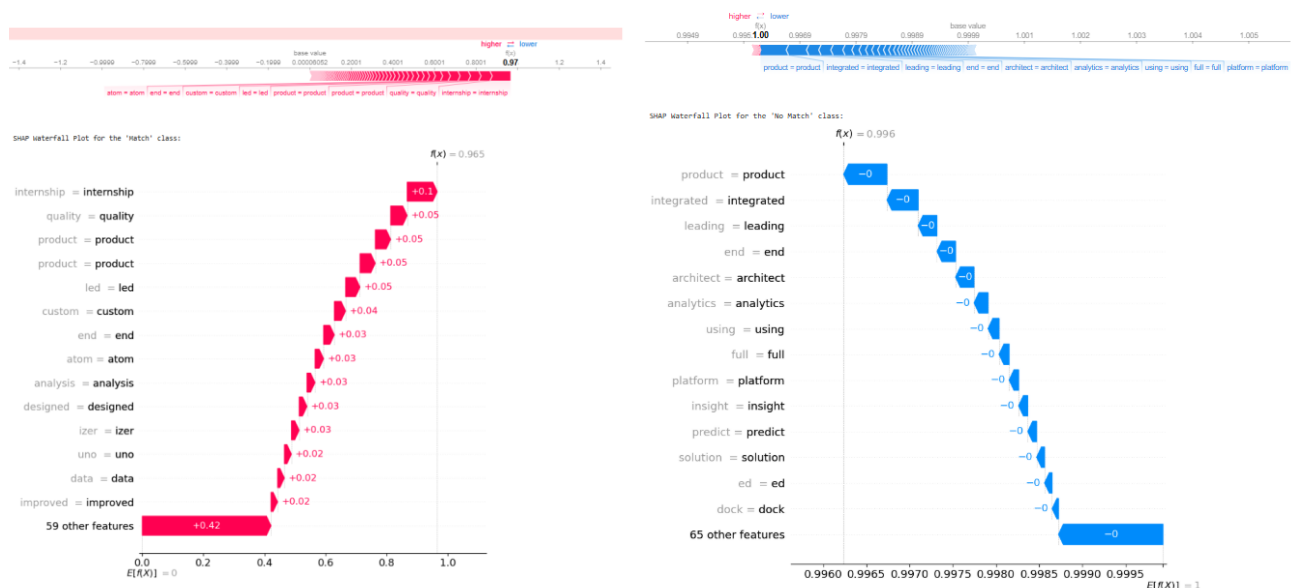


Figure 25 SHAP Explanation Generated

6. Possible Conflicts & Troubleshooting:

I. CPU Limitation:

To align with possible CPU limitations, might need to limit the CPU usage to not use multithreading, for which configured the code to use single thread using the following:

```
import os
os.environ["OMP_NUM_THREADS"] = "1"
os.environ["TOKENIZERS_PARALLELISM"] = "false"
os.environ["MKL_NUM_THREADS"] = "1"
os.environ["KMP_INIT_AT_FORK"] = "FALSE"
```

II. Library Conflict:

Conflicting version of some libraries could cause error in the environment 'streamlit_env', for which the specific version of these libraries are recommended to replicate the project to ensure smooth execution. A few lists are given below

Libraries	Version
pip	24.0
nltk	3.8.1
pandas	2.2.2
numpy	1.26.4

III. Notebook Crashing - Environment Preflight Test

To prevent system crashes, this preflight test code below should be run **before** training on pre-processed data inside the IDE in testing device. This will help to determine the optimal **model, device type**, and encoder 'max_length' that works with PC's hardware configuration and limitations.

```
bi_encoder_name = "sentence-transformers/all-MiniLM-L6-v2" # Bi-Encoder model
bi = SentenceTransformer(bi_encoder_name, device="cpu") # Chose device="cuda" if needed
bi.max_seq_length = 256 # reduce memory

# Use a 1-output head model:
ce_model_name = "cross-encoder/ms-marco-MiniLM-L-6-v2" # small & fast; single score
ce = CrossEncoder(ce_model_name, num_labels=1, device="cpu", max_length=256)

sample_resume = str(X_train.iloc[0]["resume_segment"])
sample_answer = str(X_train.iloc[0]["interview_answer"])

_ = bi.encode([sample_resume, sample_answer], show_progress_bar=False)
_ = ce.predict([(sample_resume, sample_answer)], convert_to_numpy=True)

print("Preflight OK.")
```

This configuration manual provides the necessary steps and instructions for setting up, executing, and understanding the implementation of the Interview inconsistency detection framework.

References

Google Survey link:

<https://docs.google.com/forms/d/e/1FAIpQLSehZZSICX7eH3elanDJvcUtPqNMpu5XMp33kZ2bdPhg9zSuHA/viewform?usp=sharing&oid=109136116658387132990>