**MileStone-2**

**Edu2Job Dataset**

In this process, we have loaded the dataset, performed preprocessing steps to clean and prepare the data, and included various visualizations to better understand the dataset and its features.

* Load the dataset
* Handling missing values
* Handle Outliers
* Label Encode Categorical Variables
* Feature Scaling (Standardization)
* Feature Selection (X, y)
* Train-Test Split
* Save Preprocessed Data
* Visualizations (Safe and Auto-Detecting Columns)

**Python code:**

import os

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.model\_selection import train\_test\_split

sns.set\_style("whitegrid")

**# Step 1: Load Dataset**

file\_path = r"C:\Users\prami\INFOSYS\Edu2Job\_dataset\_.csv"

if not os.path.exists(file\_path):

raise FileNotFoundError(f"❌ File not found at: {file\_path}")

df = pd.read\_csv(file\_path)

df\_original = df.copy()

print("✅ Dataset loaded successfully!")

print("Shape:", df.shape)

print("Columns:", df.columns.tolist())

**# Step 2: Handle Missing Values (Pandas 3.0 Safe)**

print("\nMissing values before cleaning:\n", df.isnull().sum())

# Separate numeric & categorical columns

num\_cols = df.select\_dtypes(include=[np.number]).columns

cat\_cols = df.select\_dtypes(exclude=[np.number]).columns

# Prepare fill values for missing data

fill\_values = {}

# Numeric columns → fill with mean

for col in num\_cols:

fill\_values[col] = df[col].mean()

# Categorical columns → fill with mode (most frequent value)

for col in cat\_cols:

if not df[col].mode().empty:

fill\_values[col] = df[col].mode()[0]

else:

fill\_values[col] = "Unknown"

# Apply fills (safe and warning-free)

df.fillna(value=fill\_values, inplace=True)

print("\n✅ Missing values handled safely (no warnings).")

**# Step 3: Handle Outliers (IQR Capping)**

def cap\_outliers(df, column):

Q1 = df[column].quantile(0.25)

Q3 = df[column].quantile(0.75)

IQR = Q3 - Q1

lower = Q1 - 1.5 \* IQR

upper = Q3 + 1.5 \* IQR

df[column] = np.where(df[column] < lower, lower,

np.where(df[column] > upper, upper, df[column]))

return df

numeric\_cols = df.select\_dtypes(include=[np.number]).columns

for col in numeric\_cols:

df = cap\_outliers(df, col)

print("\n✅ Outliers capped using IQR method.")

**# Step 4: Label Encode Categorical Variables**

label\_encoders = {}

for col in df.select\_dtypes(include='object').columns:

le = LabelEncoder()

df[col] = le.fit\_transform(df[col].astype(str))

label\_encoders[col] = le

print("\n✅ Label encoding applied successfully.")

**# Step 5: Feature Scaling (Standardization)**

scaler = StandardScaler()

df[numeric\_cols] = scaler.fit\_transform(df[numeric\_cols])

print("✅ Numeric features scaled successfully.")

**# Step 6: Feature Selection (X, y)**

target\_col = 'Job Role'

if target\_col not in df.columns:

raise KeyError(f"❌ Target column '{target\_col}' not found!")

X = df.drop(columns=[target\_col])

y = df[target\_col]

print("\nX shape:", X.shape)

print("y shape:", y.shape)

**# Step 7: Train-Test Split**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.3, random\_state=42,

stratify=y if len(y.unique()) > 1 else None

)

print("\n✅ Train-test split completed!")

print("X\_train shape:", X\_train.shape)

print("X\_test shape:", X\_test.shape)

print("y\_train distribution:\n", y\_train.value\_counts())

print("y\_test distribution:\n", y\_test.value\_counts())

**# Step 8: Save Preprocessed Data**

output\_before = r"C:\Users\prami\INFOSYS\Edu2Job\_Before\_Preprocessing.csv"

output\_after = r"C:\Users\prami\INFOSYS\Edu2Job\_After\_Preprocessing.csv"

df\_original.to\_csv(output\_before, index=False)

df.to\_csv(output\_after, index=False)

print(f"\n💾 Files saved successfully:")

print(f"→ Before preprocessing: {output\_before}")

print(f"→ After preprocessing: {output\_after}")

**# Step 9: Visualizations (Safe and Auto-Detecting Columns)**

import matplotlib.pyplot as plt

import seaborn as sns

sns.set\_style("whitegrid")

print("\n📊 Generating visualizations...")

# Clean column names (remove accidental spaces)

df\_original.columns = df\_original.columns.str.strip()

# 1️⃣ Bar Plot – Count of Job Roles

if 'Job Role' in df\_original.columns:

plt.figure(figsize=(10, 5))

sns.countplot(y='Job Role', data=df\_original, order=df\_original['Job Role'].value\_counts().index)

plt.title('Distribution of Job Roles')

plt.xlabel('Count')

plt.ylabel('Job Role')

plt.tight\_layout()

plt.show()

# 2️⃣ Bar Plot – Degree vs Job Role

if all(col in df\_original.columns for col in ['Degree', 'Job Role']):

plt.figure(figsize=(10, 5))

sns.countplot(x='Degree', hue='Job Role', data=df\_original)

plt.title('Degree vs Job Role')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

# 3️⃣ Box Plot – GPA vs Job Role

if all(col in df\_original.columns for col in ['GPA', 'Job Role']):

plt.figure(figsize=(8, 5))

sns.boxplot(x='Job Role', y='GPA', data=df\_original)

plt.title('GPA Distribution by Job Role')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

# 4️⃣ Strip Plot – Certifications vs Job Role

# (Safely handle column naming variations)

possible\_cert\_cols = [c for c in df\_original.columns if "cert" in c.lower()]

cert\_col = possible\_cert\_cols[0] if possible\_cert\_cols else None

if cert\_col and 'Job Role' in df\_original.columns:

plt.figure(figsize=(10, 5))

sns.stripplot(x=cert\_col, y='Job Role', data=df\_original, jitter=0.25, size=6)

plt.title(f'{cert\_col} vs Job Role')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

# 5️⃣ Count Plot – Industry Distribution

if 'Industry' in df\_original.columns:

plt.figure(figsize=(10, 5))

sns.countplot(y='Industry', data=df\_original, order=df\_original['Industry'].value\_counts().index)

plt.title('Industry Distribution')

plt.xlabel('Count')

plt.ylabel('Industry')

plt.tight\_layout()

plt.show()

# 6️⃣ Bar Plot – Experience Level vs Job Role

if all(col in df\_original.columns for col in ['Experience Level', 'Job Role']):

plt.figure(figsize=(10, 5))

sns.countplot(x='Experience Level', hue='Job Role', data=df\_original)

plt.title('Experience Level vs Job Role')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

# 7️⃣ Correlation Heatmap – Numeric Features

numeric\_cols = df\_original.select\_dtypes(include=[np.number]).columns

if len(numeric\_cols) > 1:

plt.figure(figsize=(8, 6))

sns.heatmap(df[numeric\_cols].corr(), annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Between Numeric Features')

plt.tight\_layout()

plt.show()

print("\n🎨 Visualization completed successfully!")

**Output:**

✅ Dataset loaded successfully!

Shape: (1000, 8)

Columns: ['Degree', 'Major', 'GPA', 'Certifications', 'Skills', 'Job Role', 'Industry', 'Experience Level']

Missing values before cleaning:

Degree 12

Major 13

GPA 10

Certifications 20

Skills 10

Job Role 10

Industry 14

Experience Level 10

dtype: int64

✅ Missing values handled safely (no warnings).

✅ Outliers capped using IQR method.

✅ Label encoding applied successfully.

✅ Numeric features scaled successfully.

X shape: (1000, 7)

y shape: (1000,)

✅ Train-test split completed!

X\_train shape: (700, 7)

X\_test shape: (300, 7)

y\_train distribution:

Job Role

9 70

1 59

3 55

11 53

2 50

12 49

8 49

10 48

7 48

13 47

0 46

4 43

6 43

5 40

Name: count, dtype: int64

y\_test distribution:

Job Role

9 30

1 25

3 23

11 23

2 22

12 21

7 21

8 21

13 20

10 20

0 19

4 19

6 19

5 17

Name: count, dtype: int64

**💾 Files saved successfully:**

→ Before preprocessing: C:\Users\prami\INFOSYS\Edu2Job\_Before\_Preprocessing.csv

→ After preprocessing: C:\Users\prami\INFOSYS\Edu2Job\_After\_Preprocessing.csv

**📊 Generating visualizations:**











