# MLVDA Final Exam - Job Market Analysis

# Bardia Bahrami Monajemi

# 1. Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import re

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

# 2. Load Dataset

df = pd.read\_csv("jobs\_dataset.csv", low\_memory=False)

print(" Dataset loaded successfully! Rows:", len(df))

# 3. Clean Columns

df.columns = df.columns.str.strip().str.lower()

df = df.loc[:, ~df.columns.str.contains('^unnamed')]

df = df.drop\_duplicates()

# 4. Handle Missing Values

df = df.fillna("Unknown")

print(" Missing values handled.")

# 5. Combine Job Type Columns

job\_cols = [c for c in df.columns if 'jobtype' in c]

if job\_cols:

df['job\_type'] = df[job\_cols].bfill(axis=1).iloc[:, 0]

else:

df['job\_type'] = "Unknown"

df['job\_type'] = df['job\_type'].str.title().replace({

'Full Time': 'Full-Time',

'Part Time': 'Part-Time'

})

# 6. Clean Salary Column

def extract\_salary(s):

if pd.isna(s):

return np.nan

text = str(s).lower()

if not any(k in text for k in ['$', 'year', 'month', 'hour']):

return np.nan

nums = re.findall(r"\d{2,7}", text.replace(",", ""))

if not nums:

return np.nan

avg = np.mean(list(map(int, nums)))

if avg < 800 or avg > 500000:

return np.nan

return avg

df['salary\_clean'] = df['salary'].apply(extract\_salary)

df = df[df['salary\_clean'].notna()]

# 7. Salary Category

df['salary\_category'] = pd.qcut(df['salary\_clean'], 3, labels=['Low', 'Medium', 'High'])

# 8. Encode Job Type for ML

le = LabelEncoder()

df['job\_type\_encoded'] = le.fit\_transform(df['job\_type'])

# 9. Text Vectorization

text\_features = ['description', 'positionname']

df['combined\_text'] = df[text\_features].astype(str).agg(' '.join, axis=1)

vectorizer = TfidfVectorizer(stop\_words='english', max\_features=500)

X\_text = vectorizer.fit\_transform(df['combined\_text'])

# 10. Numeric Features

numeric\_cols = ['salary\_clean']

X\_num = df[numeric\_cols].values

# 11. Combine Features

from scipy.sparse import hstack

from scipy import sparse

X = hstack((X\_text, sparse.csr\_matrix(X\_num)))

y = df['job\_type\_encoded']

# 12. Train/Test Split

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

X, y, test\_size=0.2, random\_state=42

)

# 13. Random Forest Model

rf = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf.fit(X\_train, y\_train)

y\_pred = rf.predict(X\_test)

# 14. Evaluation

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

print("Accuracy:", round(accuracy\_score(y\_test, y\_pred)\*100, 2), "%")

# 15. Save Cleaned Data for Tableau

keep\_cols = [

'company', 'location', 'description', 'positionname',

'salary', 'salary\_clean', 'salary\_category', 'job\_type'

]

keep\_cols = [c for c in keep\_cols if c in df.columns]

clean\_df = df[keep\_cols].drop\_duplicates()

clean\_df.to\_csv("cleaned\_jobs\_tableau\_FINAL\_PRO.csv", index=False, encoding="utf-8-sig")

print("\n Exported cleaned Tableau dataset successfully!")

print(" File: cleaned\_jobs\_tableau\_FINAL\_PRO.csv")

print(clean\_df.head(10))