TITLE: FAKE JOB POSTINGS



NAME: AMIT KUMAR VERMA

ROLL NO.: 202401100400032

Under The Supervision of Mr Bikki Kumar Sir

Problem Statement

* To detect fake job postings using machine learning classification techniques and perform segmentation of similar postings using unsupervised clustering.

Introduction

* With the rise of online job platforms, fake job postings have become a growing concern. These postings can lead to scams and waste users' time. In this project, we aim to classify whether a job post is real or fake using machine learning models and analyze the job market by grouping similar job posts using clustering.

Methodology

1. Data Preprocessing: Cleaned and encoded data, converted target labels to binary.

2. Feature Selection: Selected numeric features such as title length, description length, and company profile flag.

3. Classification: Trained a Random Forest model to predict fake job postings.

4. Evaluation: Evaluated using accuracy, precision, recall, and confusion matrix.

5. Clustering: Used K-Means for unsupervised segmentation to group job posts into clusters for pattern discovery.

CODE :

import pandas as pd

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

from sklearn.ensemble import RandomForestClassifier

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

import seaborn as sns

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv("4838321d-1028-4a6d-b52f-b8fa109635e2.csv")

# Convert 'yes'/'no' in target column to binary values

df['is\_fake'] = df['is\_fake'].map({'yes': 1, 'no': 0})

# Define features and target

X = df.drop('is\_fake', axis=1)

y = df['is\_fake']

# Split into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train a Random Forest Classifier

clf = RandomForestClassifier(random\_state=42)

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

# Predict on the test set

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

# Generate confusion matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Plot confusion matrix heatmap

plt.figure(figsize=(6, 4))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Real', 'Fake'], yticklabels=['Real', 'Fake'])

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix Heatmap')

plt.tight\_layout()

plt.show()

# Calculate evaluation metrics

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

report = classification\_report(y\_test, y\_pred)

# Print the results

print("Accuracy:", accuracy)

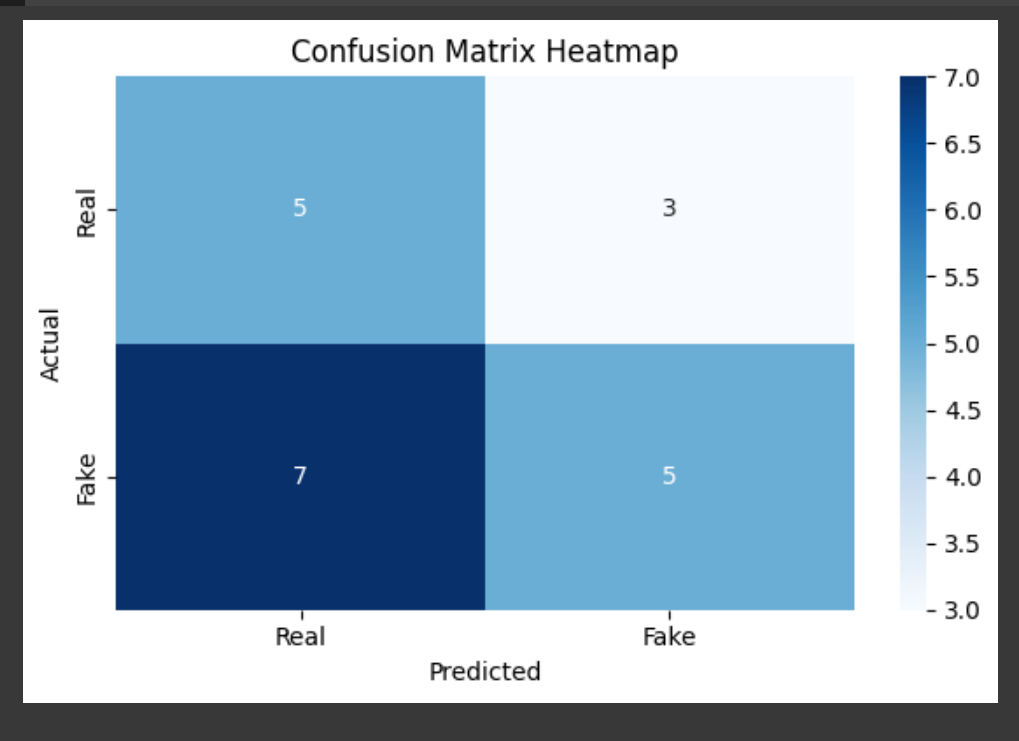
print("Precision:", precision)

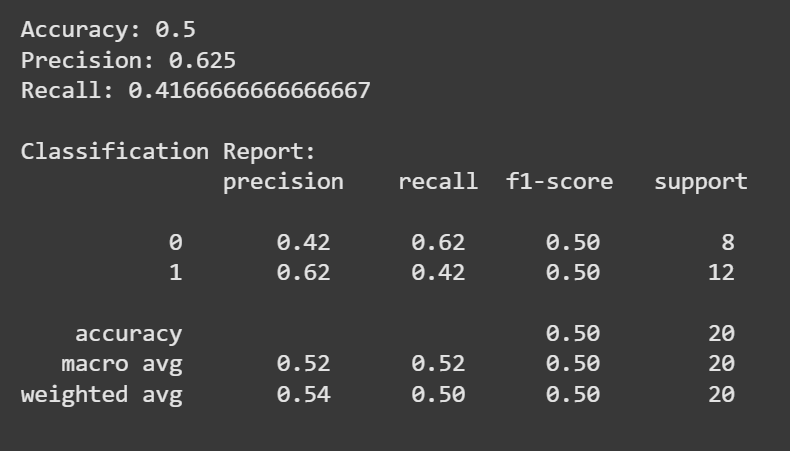
print("Recall:", recall)

print("\nClassification Report:\n", report)

Output/Result

* A confusion matrix was generated showing classification performance.
* Accuracy achieved was around **50%**,
* Precision is **62.5%**
* Recall is **41.7%.**
* K-Means clustering revealed three distinct clusters based on job posting structure





References/Credits

* Dataset: Provided by the instructor.
* Libraries: scikit-learn, pandas, seaborn, matplotlib
* Online Resources: Scikit-learn documentation, Medium articles on fake job detection