REPORT ON

Identify Fake Job Postings

By:

Ashish Srivastava

[202401100300075]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Course:

B. Tech - Computer Science and Engineering with Artificial Intelligence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Institution:

KIET Group of Institutions

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date:

22 April 2025

INTRODUCTION

Detecting fake job postings is an essential challenge in today’s online job market. Fraudulent postings can deceive job seekers, leading to financial scams, identity theft, and wasted time. To address this issue, machine learning techniques are employed to classify job postings as real or fake using key features such as job title length, description length, and the presence of a company profile.

A classification model, such as Random Forest, Logistic Regression, or Support Vector Machines (SVM), is trained using labeled job posting data. The model learns patterns and relationships among features to make predictions on unseen listings. Evaluation metrics such as **accuracy, precision, recall, and F1-score** are used to measure performance. The **confusion matrix** provides insight into misclassified entries, and its heatmap visualization helps in understanding classification trends.

Additionally, **segmentation and clustering** techniques like **K-Means** can further analyze job postings, grouping similar listings based on feature similarities. This allows for deeper insights into common characteristics of fraudulent postings, helping refine fraud detection algorithms.

The application of these machine learning methodologies enhances job market transparency, safeguards job seekers from scams, and improves hiring processes. By implementing such models, platforms can proactively filter out deceptive job listings, ensuring a trustworthy environment for employers and candidates alike.

METHODOLOGY

Detecting fake job postings involves analyzing textual features and applying machine learning techniques to classify listings as **real or fake**. The methodology consists of the following steps:

**1. Data Preprocessing**

* Load the dataset and clean the data by removing inconsistencies.
* Convert categorical features (is\_fake, has\_company\_profile) into numerical values using **Label Encoding**.
* Normalize numerical features (title\_length, description\_length) using **StandardScaler** to improve model performance.

**2. Feature Engineering**

* Analyze correlations between text-based features and job posting authenticity.
* Apply **Exploratory Data Analysis (EDA)** using histograms and scatter plots to observe trends.

**3. Model Training & Selection**

* Split data into **training and testing sets** using train\_test\_split.
* Choose classification models such as **Random Forest, Logistic Regression, or Support Vector Machines (SVM)**.
* Train the model on labeled data and evaluate predictions.

**4. Evaluation Metrics**

* Calculate **accuracy, precision, recall, and F1-score** for performance assessment.
* Generate a **confusion matrix** to analyze classification errors.

**5. Visualization with Heatmap**

* Use **Seaborn** to create a **confusion matrix heatmap**, highlighting model performance.

**6. Clustering for Fraud Detection**

* Apply **K-Means clustering** to group similar job postings, helping to uncover fraudulent patterns.

CODE

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

from sklearn.cluster import KMeans

# Load dataset

df = pd.read\_csv("C:\\Users\\Ashish\\Desktop\\ai\\fake\_jobs.csv")

# Convert 'is\_fake' column to numeric (yes = 1, no = 0)

label\_encoder = LabelEncoder()

df["is\_fake"] = label\_encoder.fit\_transform(df["is\_fake"])

# Feature selection

X = df[["title\_length", "description\_length", "has\_company\_profile"]]

y = df["is\_fake"]

# Train-test split

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

# Normalize numerical features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Train classification model (Random Forest)

clf = RandomForestClassifier(*random\_state*=42)

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

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

# Calculate evaluation metrics

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

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

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

f1 = f1\_score(y\_test, y\_pred)

print(*f*"Accuracy: {accuracy*:.2f*}")

print(*f*"Precision: {precision*:.2f*}")

print(*f*"Recall: {recall*:.2f*}")

print(*f*"F1 Score: {f1*:.2f*}")

# Generate confusion matrix heatmap

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

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

sns.heatmap(cm, *annot*=True, *fmt*='d', *cmap*='Blues')

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix Heatmap")

plt.show()

# Perform clustering using K-Means (segmentation)

kmeans = KMeans(*n\_clusters*=2, *random\_state*=42)

df["cluster"] = kmeans.fit\_predict(X)

# Visualize clusters

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

sns.scatterplot(*x*=df["title\_length"], *y*=df["description\_length"], *hue*=df["cluster"], *palette*="Set1")

plt.xlabel("Title Length")

plt.ylabel("Description Length")

plt.title("Job Posting Clustering")

plt.show()

SCREENSHORT



