Identify Fake Job Postings

Use job post text features to classify whether a posting is real or fake.

K.I.E.T. Group of Institutions

Ghaziabad

NAME – MAHI

BRANCH – CSEAI

SECTION - B

ROLL NO – 202401100300147

CLASS ROLL NO. – 72

INTRODUCTION

Identifying Fake Job Postings

Problem Overview:
With the rise of online job boards and recruitment platforms, fake job postings have become a significant concern. These fraudulent job advertisements often deceive job seekers by offering too-good-to-be-true opportunities or asking for personal information or payment. This can lead to wasted time, financial loss, identity theft, and frustration for job seekers.

Key Issues:

- **Deceptive Job Offers**: Fake postings may promise unrealistic salaries, flexible work-from-home opportunities, or immediate hiring without qualifications.
- **Scams and Fraud**: Fraudsters use job ads to collect sensitive information (e.g., Social Security numbers,

bank details) or request money upfront for supposed "training" or "processing fees."

- **Lack of Information**: Legitimate job postings typically include detailed company information, job descriptions, and contact details. Fake ads often lack these crucial details or provide vague, misleading information.

Consequences:

- **Job Seekers**: Wasted time applying for jobs that don't exist, financial scams, and a loss of trust in online job platforms.
- **Reputation Damage**: Job platforms and legitimate companies risk their reputation when fake job postings are not filtered out, losing user trust.

Goal: The goal is to create an automated solution that identifies and flags fake job postings, helping job seekers avoid scams and ensuring the reliability of job platforms.

METHODOLOGY

Methodology for Identifying Fake Job Postings

1. **Data Collection**: Gather a dataset of job postings with labels indicating if the posting is real or fake. This includes features like job title, description, company profile, and more.

2. **Data Preprocessing**: Clean the data by handling missing values, and preprocess text features (e.g., job titles/descriptions) using techniques like tokenization, stemming, and vectorization.

3. **Feature Engineering**: Extract relevant features such as job title length, description length, presence of company profile, and keywords to help distinguish real from fake posts.

4. **Model Selection & Training**: Choose a machine learning model (e.g., Logistic Regression, Random Forest) and train it on the dataset to classify job postings as real or fake.

5. **Model Evaluation**: Assess model performance using metrics like accuracy, precision, recall, and F1-score.

Fine-tune the model using cross-validation and hyperparameter optimization.

6. **Deployment**: Deploy the trained model to classify new job postings in real-time, ensuring automatic detection of fake ads.

7. **Post-Deployment**: Continuously monitor and retrain the model with new data to maintain its effectiveness and accuracy in detecting fake job postings.

CODE

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Load the CSV file (adjust path accordingly)
file_path = 'fake_jobs.csv'  # Replace with your CSV file path
```

```
df = pd.read_csv(file_path)

# Display the first few rows to check data structure
print(df.head())

# Assume the columns are:
# title_length - Length of the job title
# description_length - Length of the job description
# has_company_profile - Whether the job has a company profile (1 = yes, 0 = no)
# is_fake - Target variable (1 = fake, 0 = real)

# Define features (X) and target (y)
X = df[['title_length', 'description_length', 'has_company_profile']]
y = df['is_fake']

# Split data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a Logistic Regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

```
# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model performance
print("Classification Report:\n", classification_report(y_test, y_pred))

# Plot the confusion matrix
cm = confusion_matrix(y_test, y_pred)
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')
plt.show()
```

```
plt.title('Confusion Matrix')
plt.show()
```

```
   title_length  description_length  has_company_profile is_fake
0            72                 740                    1     yes
1            95                 476                    0      no
2            60                 662                    1     yes
3            34                 317                    0      no
4            67                 884                    0     yes
Classification Report:
              precision    recall  f1-score   support

          no       0.56      0.62      0.59         8
         yes       0.73      0.67      0.70        12

    accuracy                           0.65        20
   macro avg       0.64      0.65      0.64        20
weighted avg       0.66      0.65      0.65        20
```


```
weighted avg       0.66      0.65      0.65        20
```