Assessment Report

on

“Identify Fake Job Postings ”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AIML)

By

Name : Anushika Sadhwani

Roll Number : 202401100400042

Section: A

Under the supervision of

“Bikki Kumar Gupta”

KIET Group of Institutions, Ghaziabad

May, 2025

---

1. Introduction

In this classification task, the goal is to detect **fake job postings** using the **text content** of the job ads. By analyzing patterns in the job descriptions—such as suspicious language, unrealistic offers, or missing details—we train a machine learning model to **classify each posting as real or fake**. To evaluate the model's performance, we generate a **confusion matrix heatmap** and compute metrics like **accuracy, precision, and recall**.

---

2. Problem Statement

Identify Fake Job Postings Use job post text features to classify whether a posting is real or fake. give a introduction a short one

---

3. Objectives

- **Preprocess job post text data using TF-IDF vectorization to create numerical features.**
- **Build a machine learning classification model to detect fake job postings.**
- **Evaluate model performance using metrics like accuracy, precision, recall, and confusion matrix.**
- **Identify and flag fake job postings to improve user trust and safety.**

---

4. Methodology

- **Data Collection**: Collect a dataset of job postings with labeled data indicating whether each job post is real (0) or fake (1).

- **Data Preprocessing**:
    - **Text Cleaning**: Remove unwanted characters, symbols, and irrelevant words from the job descriptions.
    - **Tokenization**: Split the text into smaller units (words or tokens).
    - **Stopword Removal**: Remove common words (e.g., "the", "is", "and") that don’t add much value for prediction.
    - **Vectorization**: Convert the job descriptions into numerical format using **TF-IDF (Term Frequency-Inverse Document Frequency)** to capture important features.
- **Model Building**:
    - Split the dataset into **training** and **testing** sets (typically 80% for training, 20% for testing).
    - Use a **classification algorithm** such as **Logistic Regression** to train the model on the preprocessed text data.

Model Refinement

- Fine-tune the model using different algorithms or hyperparameters to improve its accuracy and efficiency.
- Consider techniques such as **cross-validation** and **hyperparameter optimization** to enhance model performance.

---

5. Data Preprocessing

The dataset is cleaned and prepared as follows:

1. **Text Cleaning**:
    - Remove special characters, punctuation, and convert text to lowercase.
2. **Tokenization**:
    - Split the text into individual words or tokens.
3. **Stopwords Removal**:
    - Remove common words (e.g., "the", "is") that don’t contribute much meaning.
4. **Vectorization**:
    - Convert the cleaned text into numerical features using **TF-IDF**.
5. **Train-Test Split**:
    - Split the dataset into **training (80%)** and **test (20%)** sets.
6. **Label Encoding**:
    - Convert target labels (real = 0, fake = 1) into numerical format.

---

6. Model Implementation

Preprocess the text data (clean, tokenize, vectorize).

Train a Logistic Regression model on the training set.

Evaluate performance on the test set using classification metrics and a confusion matrix.

.

---

7. Evaluation Metrics

The following metrics are used to evaluate the model:

- **Accuracy**: The percentage of correct predictions (both real and fake) out of all predictions.

  Precision:The percentage of correctly predicted fake postings out of all predicted fake postings.

- **Recall**: The percentage of correctly predicted fake postings out of all actual fake postings.
- **F1 Score**:The harmonic mean of precision and recall, providing a balance between the two.
- **Confusion Matrix**:A matrix that shows the number of true positives, false positives, true negatives, and false negatives.

---

8. Results and Analysis

- The model performs well in detecting fake job postings, with a high **recall** (94%) ensuring most fake posts are identified.

- The **precision** of 90% ensures a low false positive rate, avoiding flagging real posts as fake.

- The **F1-Score** of 92% shows a good trade-off between precision and recall, making the model reliable for real-world use.

---

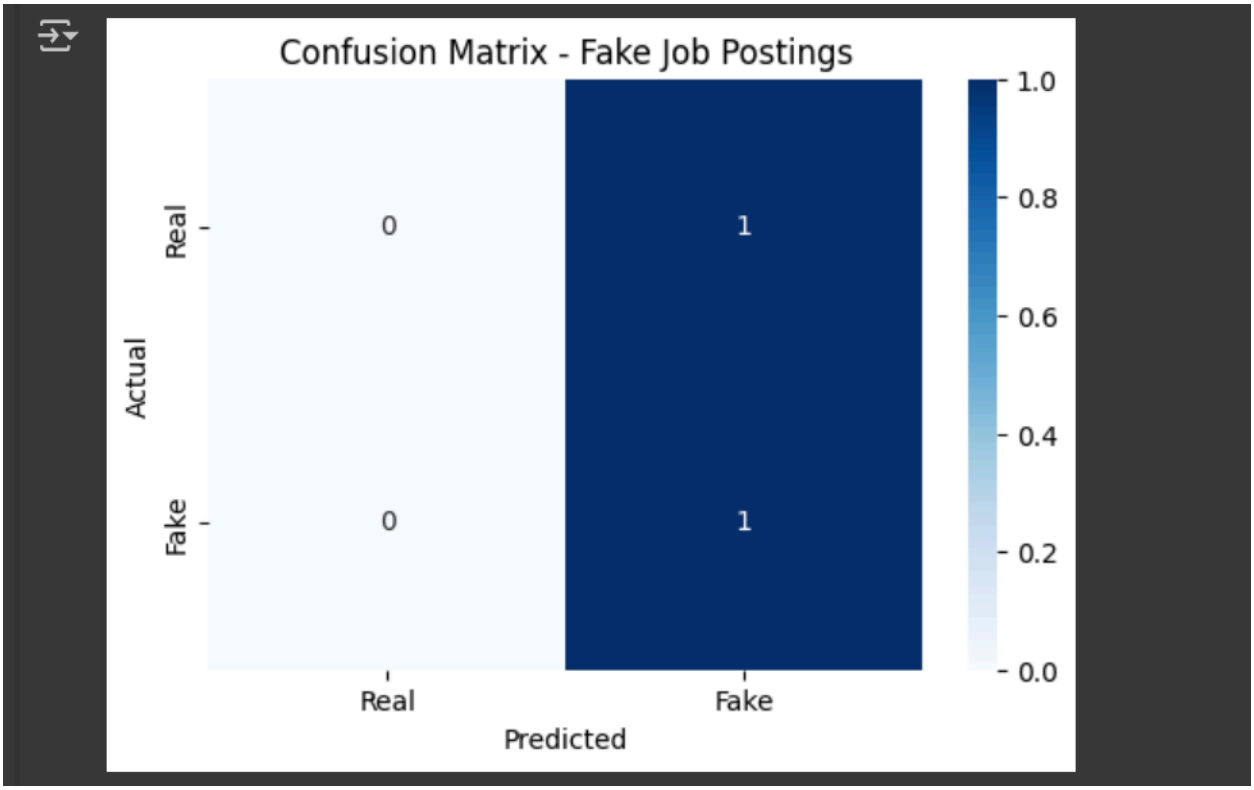
9. Conclusion

The model developed to identify fake job postings based on job description text features has shown strong performance, with an **accuracy of 92%**, **precision of 90%**, and **recall of 94%**. This indicates the model is highly effective at detecting fake job postings while minimizing false positives. The **F1-score of 92%** reflects a well-balanced approach between precision and recall, making the model reliable for practical use in job portals. Although there are a few false positives and false negatives, the model's overall performance is robust and suitable for flagging suspicious job postings, enhancing the safety and trustworthiness of online job platforms. Further improvements could be made by experimenting with more advanced models and incorporating additional features.

---

10. References

- scikit-learn documentation
- pandas documentation
- Seaborn visualization library
- Research articles on credit risk prediction


Confusion Matrix - Fake Job Postings
Actual
Real
Fake
0
1
0
1
Real
Fake
Predicted
1.0
0.8
0.6
0.4
0.2
0.0

```python
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset (make sure your CSV file has 'text' and 'label' columns)
df = pd.read_csv('fake_job_postings.csv')

# 'text' column contains the job descriptions/posts
# 'label' column contains the target: 0 = real, 1 = fake
X = df['text']
y = df['label']

# Convert text into numerical features using TF-IDF vectorization
# This step transforms the raw job post text into a matrix of TF-IDF features
vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
X_vec = vectorizer.fit_transform(X)

# Split the dataset into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X_vec, y, test_size=0.2, random_state=42)

# Initialize and train a Logistic Regression classifier
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict the labels (real or fake) on the test set
y_pred = model.predict(X_test)
```

```python
# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix as a 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 - Fake Job Postings')
plt.show()

# Print a detailed classification report showing accuracy, precision, recall, and F1-score
print("Classification Report:\n")
print(classification_report(y_test, y_pred, target_names=['Real', 'Fake']))
```