Fake Job Posting Detection Using Machine Learning & Deep Learning

Overview

This work is based on detecting fake job posting by application of NLP and ML and DL techniques. Fake job ads are rife on the web and they cause people to commit fraud, scams, or even put their personal data at risk. The overall aim is to classify the job posting automatically as either Real or Fake using the text.. Traditional ML models and deep learning based on ANN are combined to get the best possible result with high accuracy and reliability. A Flask web application is provided to detect the fake jobs in real-time.

Dataset Sources

1. Fake Job Posting Dataset Source:

Kaggle - Real or Fake Job Posting Prediction
Size: ~18,000 job postings
Characteristics: Title, Company Profile, Description, Requirements, Benefits, etc.
Target Variable: fraudulent → (0 = Real, 1 = Fake)

2.Job Train Dataset Source:

Source: Custom compiled dataset
Size: Approximately 10,000 job postings
Goal: Test model generalization on new data

Data Preprocessing

Data Cleansing: Duplicates were removed and missing values are handled
Text Preprocessing: pool several text fields (title, description, requirements, company profile)
Text Normalization:

- Convert text to lowercase
- Remove punctuation, symbols, special characters.

Stopword Removal: With NLTK Stopword Corpus
Lemmatization and Tokenization: Reducing words to their root form
Feature Extraction:

- TF-IDF
- max_features = 5000
- ngram_range = (1,2)
- Train-Test Split: 80% training, 20 % test
- Final dataset size: ~15,000–20,000 cleaned samples.

Models Implemented

Model	Type	Description
Naive Bayes (Multinomial)	ML	A lightweight and fast probabilistic classifier, ideal for text classification.
Logistic Regression	ML	Linear model with L2 regularization; interpretable coefficients for insights.
SVM (Linear Kernel)	ML	Powerful classifier suitable for high-dimensional datasets with clear margins.
ANN (Keras Sequential)	DL	Feedforward neural network with hidden layers, ReLU activation, and dropout regularization.

Artificial Neural Network Architecture

Input Layer: TF-IDF feature vector

Hidden Layers:

- Dense (128 neurons, ReLU)
- Dropout (0.3)

Output Layer: Dense (1 neuron, Sigmoid activation)

Optimizer: Adam (learning rate = 0.001)

Loss Function: Binary Cross-Entropy

Metrics: Accuracy, AUC

Training Setup

- Frameworks Used: scikit-learn, TensorFlow, Keras, NLTK, Matplotlib, Seaborn
- Validation Strategy: 5-Fold Cross Validation
- Batch Size: 32
- Epochs: 5

Performance Comparison

Job Train Dataset

Model	Accuracy	Precision	Recall	F1-Score	AUC
ANN (Keras)	97.31%	0.7945	0.6374	0.7073	0.9359
SVM (Linear)	96.92%	0.7143	0.6593	0.6857	0.8749
Logistic Regression	96.03%	0.5962	0.6813	0.6359	0.9441
Naive Bayes	90.49%	0.3290	0.8352	0.4720	0.9329

Fake Job Posting Dataset

Model	Accuracy	Precision	Recall	F1-Score	AUC
ANN (Keras)	98.40%	0.9143	0.7399	0.8179	0.9877
SVM (Linear)	98.29%	0.8146	0.8382	0.8262	0.9751
Logistic Regression	97.48%	0.6948	0.8555	0.7668	0.9839
Naive Bayes	91.55%	0.3543	0.9133	0.5105	0.9747

Visualizations

- **Confusion Matrices:** Fewer false positives for ANN

- **ROC Curves:** ANN achieved AUC > 0.98 on both datasets
- **Performance Plots:** Precision, Recall, F1-score comparison

Results & Insights

- **Best Model:** ANN (Keras)
- **Highest Accuracy:** 98.4%
- **Best AUC:** 0.9877
- **Key Insight:** TF-IDF + ANN pipeline captures complex textual relationships effectively
- **Generalization:** Performs strongly across multiple datasets

Conclusion

This work is a solid proof of concept for fighting job posting fraud with AI and NLP.🔍 Key Takeaways ANN achieves the best performance with ~98% accuracy comparing with other traditional ML methods TF-IDF can be a strong text representation for text classification Flask web application to detect fake jobs in real time
Methodology is extendable to other fields such as scam e-mail detection or fake news classification

References

1. Kaggle Dataset: Real or Fake Job Posting Prediction
2. Scikit-learn Documentation
3. Keras API – TensorFlow
4. Bird, Klein & Loper (2009) – *Natural Language Processing with Python*
5. Zhang, Y. & Wallace, B. (2015) – *CNN Sensitivity Analysis for Sentence Classification*