

University of Illinois Springfield

Final Project Report

Real / Fake Job Posting Prediction Data

Nancy Bhargava

Kriti Jain

Introduction to Machine Learning

Elham Khorasani Buxton

Table of Contents

Abstract	3
Problem Definition and Project Goals	3
Related Work.....	4
Data Exploration & Preprocessing.....	5
Exploratory Data Analysis	5
Feature Engineering and Processing	7
Significance Testing	8
Data Scaling and Train-Test Split	8
Data Modeling and Results	8
Multi-Layer Perceptron (Neural Network)	9
Random Forest Regressor	Error! Bookmark not defined.
eXtreme Gradient Boosting Machine	Error! Bookmark not defined.
Geo-Spatial Clustering	Error! Bookmark not defined.
Conclusion	10
References.....	12

Abstract

With the increase in employment scams, particularly exacerbated by economic downturns such as the COVID-19 pandemic, the need to combat fraudulent job postings is more pressing than ever. Scammers prey on individuals seeking employment by offering enticing job opportunities and then exploiting them for personal information or money. As a university student, I have personally encountered such scams via email. To address this issue, this project utilizes Machine Learning techniques to detect and classify fraudulent job postings. The dataset, obtained from Kaggle, comprises features defining job postings categorized as real or fake. While fake job postings constitute a small fraction of the dataset, this project aims to develop models capable of accurately identifying and flagging them to protect job seekers from falling victim to scams.

Problem Definition and Project Goals

The problem definition is to develop a classification machine learning model that can accurately distinguish between real and fraudulent job postings. The data comprises of both numeric as well as categorical variables and we must extract the important variables and use them for our modelling. We have created a benchmark model to explore the simple statistic accuracy of the model and then predict using other binary classification models and Neural Network. Originally, the data had the following variables, data types and description:

Variable Name	Data Type	Description
job_id	int	Identification number given to each job posting
title	text	A name that describes the position or job
location	text	Information about where the job is located
department	text	Information about the department this job is offered by
salary_range	text	Expected salary range
company_profile	text	Information about the company
description	text	A brief description about the position offered
requirements	text	Pre-requisites to qualify for the job

benefits	text	Benefits provided by the job
telecommuting	boolean	Is work from home or remote work allowed
has_company_logo	boolean	Does the job posting have a company logo
has_questions	text	Does the job posting have any questions
employment_type	text	5 categories – Full-time, part-time, contract, temporary and other
required_experience	text	Can be – Internship, Entry Level, Associate, Mid-senior level, Director, Executive or Not Applicable
required_education	text	Can be – Bachelor's degree, high school degree, unspecified, associate degree, master's degree, certification, some college coursework, professional, some high school coursework, vocational
Industry	text	The industry the job posting is relevant to
Function	text	The term to determining a job's functionality
Fraudulent	boolean	The target variable -> 0: Real, 1: Fake

This dataset contains 18K job descriptions and is highly imbalanced with 9868 (93% of the jobs) being real and only 725 or 7% of the jobs being fraudulent. We have tried to sample our data because of its imbalanced nature and huge memory size. After cleaning and exploring our data, we have tried to extract some importance features through statistics and visualizations techniques and later concatenated to form it a document term matrix. This is a basic step for a Natural Language Processing problem.

Related Work

This dataset was shared on Kaggle, a popular platform for data enthusiasts, and it was last updated in 2020. Many people have worked on this dataset, not just for building machine learning models, but also for exploring the data and creating visualizations. However, for this project, we didn't look at the work others had done before completing our project. This was

intentional, so we could approach the problem with a fresh perspective and avoid duplicating ideas. The ideas and concepts we have explored are all in Python language. We used the references to understand the process.

A submission on [Reference\[1\]](#) mainly focused on the exploratory data analysis of this dataset. The author thoroughly explores and handle the missing values, plots the target variables, get the summary statistics for numerical features. He then focusses on the pre-processing steps by applying Part-of-Speech tagging and CountVectorizer.

The next submission on [Reference\[2\]](#) explains the wordcloud and feature importance by extracting and plotting the graphs for the different types of categories in a few variables.

Reading a few discussions like [Reference\[3\]](#) helped us understand the different ways to handle our imbalance dataset.

Data Exploration & Preprocessing

The dataset consists of a total of 17,880 rows and 18 columns including the target feature. We have sampled the 2000 rows due to memory issues. A basic summary tells us that we have an imbalanced dataset, around 13 categorical variables, 4 numerical and about 11,883 total null values. Correlation matrix was used for features: department, salary_range and function, and due to their very small values, we removed them as they were weak features. Handled other categorical missing values by replacing them with an empty string (" "). Job_id variable doesn't help us to understand if the job is fraudulent or not. It just acts as a unique identifier in terms of database term and thus we removed it too. After plotting the graphs for each variable's count and with respect to the fraudulent feature, we are left with - title, company_profile, description, requirements, benefits, industry, and we concatenated them into a single column named text. Our pre-processing included cleaning the corpus, creating wordclouds and document term matrix and creating frequency terms using our document term matrix data.

Exploratory Data Analysis

The graph in Figure 1 plot shows our imbalanced data after sampling the 2000 rows.

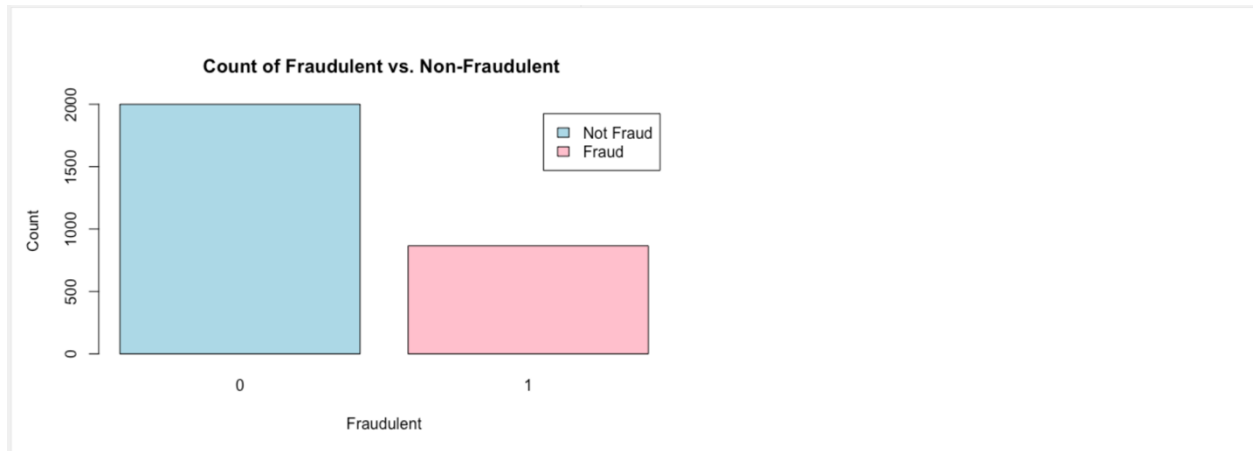


Figure 1 – Count of Fraudulent vs non-fraudulent

The null values are shown for each feature in Figure 2.

job_id	title	location	department	salary_range	company_profile	description	requirements	benefits
0	0	54	1804	2337	907	0	457	1186
telecommuting	has_company_logo	has_questions	employment_type	required_experience	required_education	industry	function.	fraudulent
0	0	0	619	1237	1377	824	1081	0
location	salary_range	company_profile	requirements	benefits	employment_type	required_experience	required_education	industry
1.884159	81.542219	31.646895	15.945569	41.381717	21.598046	43.161200	48.046057	28.750872
function.								
37.718074								

Figure 2 – Percentage of Null Values

Below are a few graphs to show each category count in these features:

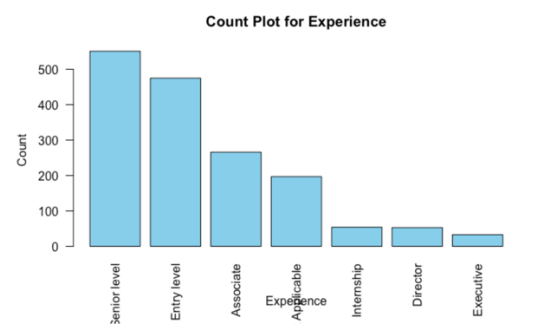


Figure 3 – Experience

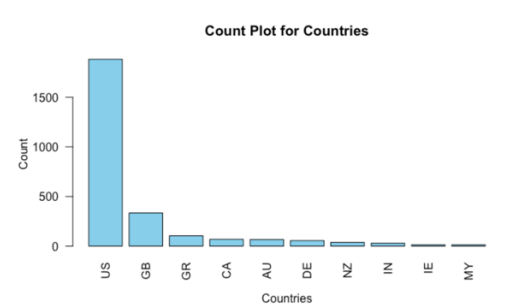


Figure 4 – Country extracted from Location

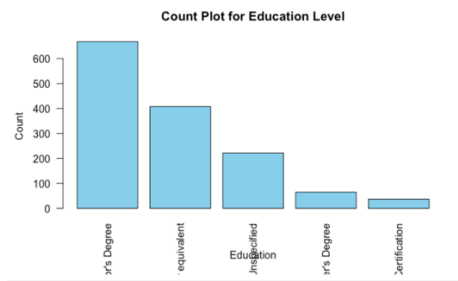


Figure 5 – Education

The most important features among these text variables of our database using chi-square test statistics.

	variable <chr>	chi_sq_statistic <dbl>
4	description	2866.00000
1	title	2744.47988
5	requirements	2398.37346
3	company_profile	1959.00000
6	benefits	1672.05447
10	industry	724.68046
11	has_company_logo	650.12294
2	country	391.92066
9	required_education	174.47519
8	required_experience	60.82870

1-10 of 11 rows

Previous 1 2 Next

Figure 6 – Feature Importance as per chi-square statistic

Feature Engineering and Processing

After discarding all the un-necessary variables, we are left with features: title, company_profile, description, requirements, benefits, industry and thus, after concatenating these variables into text feature, now we have two columns to deal with: text and the target column named as fraudulent. We now have to create a text_corpus, clean it and create Wordclouds as seen below in figure 7 and 8 respectively for non-fraudulent and fraudulent classes.



Figure 7 – Non-Fraudulent Word Cloud



Figure 8 – Fraudulent Word Cloud

Document Term Matrix was created, and the train-test split was performed on it with 70-30 ratio.

Significance Testing

For numerical variables, Correlation Matrix was created and values were extracted ranging between -1 to 1.

For categorical variables, Chi-Square test was used as our target variable was binary classification.

Data Scaling and Train-Test Split

Our dataset didn't require any scaling, but creation of document term matrix was the most important part as it consists of text data. The data was split in to training and testing set with an 90-10 ratio respectively at the beginning and for neural network purpose, our train data was split in to 70:30 ration for train and validation set.

Data Modeling and Results

The Models we chose and their respective results:

1. First of all, we have created a simple benchmark model giving us 71.8% accuracy metrics.
2. The simplest classification model: Naïve Bayes gave us 84.3% accuracy.
3. Decision Trees gave a slightly better accuracy of 86%.
4. ANN, The Neural Network gave us 93%

Artificial Neural Network (Neural Network)

They mimic the human brain's neural networks, processing input data through layers of interconnected nodes to make predictions. During training, the network adjusts its parameters to minimize prediction errors. ANNs are versatile and can handle complex data, making them suitable for tasks like job posting classification.

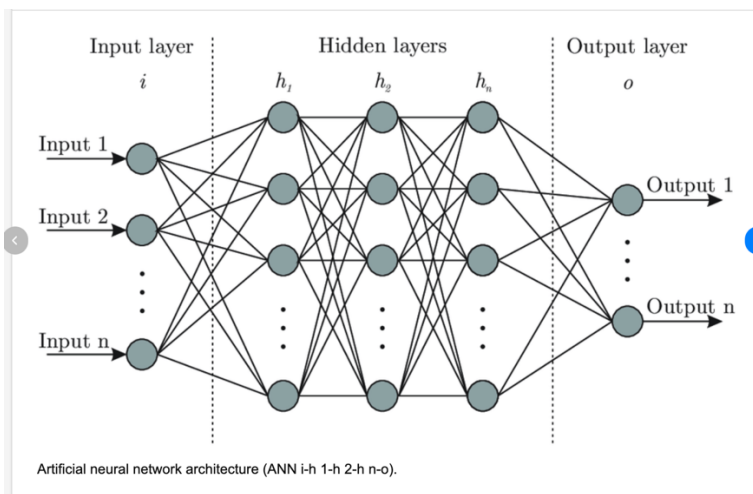


Figure 9 – Neural Network structure

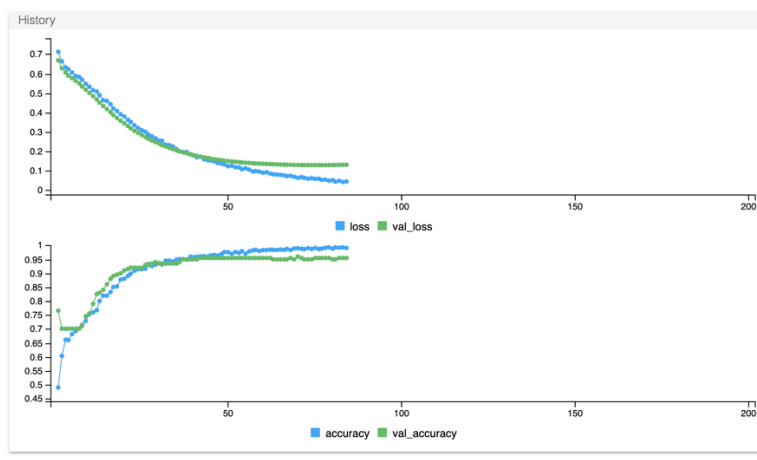


Figure 10 – Best Run ANN

Model: "sequential"		
Layer (type)	Output Shape	Param #
dense_2 (Dense)	(None, 16)	53296
dropout_1 (Dropout)	(None, 16)	0
dense_1 (Dense)	(None, 16)	272
dropout (Dropout)	(None, 16)	0
dense (Dense)	(None, 1)	17
Total params: 53585 (209.32 KB)		
Trainable params: 53585 (209.32 KB)		
Non-trainable params: 0 (0.00 Byte)		

Figure 11 – Parameters for the best Run

Conclusion

In conclusion, three types of models, Naïve bayes, Decision Trees and ANN were trained excluding the benchmark model over the dataset to compare their performances to predict if the job posting is a fake or real. The data features after transformation, was not compatible for Logistic regression, GBM and Random Forest and were taking a lot of memory and time, we chose to not include them. As this was the binary classification model, the models were compared using the Accuracy and F-1 Score metrics calculated on the predictions on the test set.

Model Name	Accuracy	Precision	Recall	F1-Score
Naive Bayes Model	0.8430233	0.8317152	0.9431193	0.8839209
Decision Trees	0.8546512	0.7479339	0.7387755	0.7433265
ANN	0.9348	0.8677	0.8974	0.8823

Figure 12 – Comparison of models

Figure 12 shows the comparison of all the metrics of the model.



About our best Model: ANN

This code sets up and trains a neural network model to classify job postings as real or fake. It begins by defining parameters such as learning rate, number of nodes, and batch size. The 'keras' library is then loaded to facilitate model creation. The neural network architecture is established with two hidden layers, each consisting of 32 nodes and utilizing the sigmoid activation function. Dropout layers are incorporated to mitigate overfitting, and an output layer with one neuron and a sigmoid activation function is added for binary classification. During training, the Adam optimizer adjusts model parameters to minimize the binary crossentropy loss, with early stopping implemented to prevent overfitting. The model is trained over 200 epochs using training data, with validation data used for performance monitoring. Overall, this process aims to develop an effective neural network model for distinguishing between real and fraudulent job postings.

Run	
context	local
script	tun.R
started	2024-05-06 02:24:49.27288 GMT
time	00:01:46
Metrics	
loss	0.0449
accuracy	0.9911
val_loss	0.1313
val_accuracy	0.9552
Flags	
learning_rate	0.0001
nodes	16
batch_size	16
activation	sigmoid
Optimization	
loss	binary_crossentropy
optimizer	<keras.src.optimizers.adam.Adam object at 0x0000014F3566FA90>
lr	0.0001
Training	
epochs	84/200

Figure 13 – Hyperparameters

References

- [1] SEIF WAEL, "Real/Fake Jobs  | EDA & Modelling  | 99%" [Online]Available:
<https://www.kaggle.com/code/seifwael123/real-fake-jobs-eda-modelling-99#Models>.
- [2] Baby Shree J, " **Fake_Real_Job** " [Online]. Available:
<https://www.kaggle.com/code/babyshree/fake-real-job>.
- [3] Shivam Bansal, " **Real / Fake Job Posting Prediction**" [Online]. Available:
<https://www.kaggle.com/datasets/shivamb/real-or-fake-fake-jobposting-prediction/discussion/143143>.
- [4] Chatgpt for code syntax for R