Project Report

On

Text Classification for Real and Fraudulent Job Posting

CS6120 – Natural Language Processing

Submitted by: Group 12

Team Members

Mrudula Challagonda (002765266)

Sai Shyam Nidadavolu (002792342)

Ratna Manjeera Grandhi (002738068)

Term and Year: Fall 2023

Submitted to: Prof. Uzair Ahmad

Submitted Date: December 10, 2023

Text Classification for Real and Fraudulent Job Posting

Project Setting:

The rise of online job listings has brought about new challenges in identifying fraudulent job descriptions, which can have significant consequences for job seekers. Job seekers across all population are particularly vulnerable to deceptive job postings. These individuals invest their time, resources, and hopes into finding employment, making them more prone to exploitation. This report presents a detailed analysis and implementation of a classification model to distinguish between real and fraudulent job descriptions.

Problem Definition:

The primary problem at hand is to develop a classification model that can automatically classify job descriptions as either genuine or fraudulent. This model will be valuable for job platforms, helping them identify and remove deceptive job listings, thus enhancing the user experience and safety. The specific goals of this project include:

1. Classification Model: Develop a robust classification model that uses both text data features and meta-features to predict whether a job description is real or fraudulent.

2. Feature Analysis: Identify the key textual traits, words, entities, and phrases that are indicative of fraudulent job descriptions.

3. Contextual Similarity: Implement a contextual embedding model to find job descriptions with similar content.

4. Exploratory Data Analysis: Conduct an exploratory data analysis (EDA) to uncover interesting insights within the dataset.

Methodology

We implemented the below techniques to reach our end goal:

Vectorization:

To prepare the textual data for the classification model, we will implement text vectorization techniques such as TF-IDF, Count Vectorizer and BERT embeddings. This process will convert the job descriptions into numerical representations that can be used by the machine learning model.

Text Classification using Keras:

We want to develop an ML model using the Keras library that will classify job description. The model will take into account both the text data features and the meta-features to make predictions. Techniques such as Naïve Bayes, convolutional neural networks (CNNs)

Feature Analysis:

After building the classification model, we will perform feature importance analysis to identify the key traits, words, entities, and phrases associated with fraudulent job descriptions. This will help in understanding the textual indicators of fraud.

Contextual Embedding Model:

A contextual embedding model (BERT) will be employed to find job descriptions that are most similar in content. This can help job platforms detect similar fraudulent listings even if they are not identical in wording.

Exploratory Data Analysis (EDA):

To extract meaningful insights from the dataset we will then perform EDA. That is, exploring the distribution of real and fake job descriptions, analyzing the impact of meta-features on fraud detection, and identifying patterns in the data.

Data Source and Data Description:

The dataset consists of 18,000 job descriptions, a mixture of real and fake, along with meta-information about the jobs. The meta-information may include details like job category, location, company size, and more. The key attributes of the dataset are as follows:

Text Data Features: These are the job descriptions themselves, comprising the textual information that will be the primary basis for classification.

Meta-Features: These encompass any additional data related to the jobs, which can be used to augment the classification model. This information might include job category, location, company information, and posting date.

Below is a snippet of the data:

```
[ ] df=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/MS/NLP/Project/fake_job_postings.csv')
```

```
[ ] df
```

	job_id	title	location	department	salary_range	company_profile	description	requirements	benefits	telecommuting	has_company_logo	has_ques
0	1	Marketing Intern	US, NY, New York	Marketing	NaN	We're Food52, and we've created a groundbreaki...	Food52, a fast-growing, James Beard Award-winn...	Experience with content management systems a m...	NaN	0	1	
1	2	Customer Service - Cloud Video Production	NZ, , Auckland	Success	NaN	90 Seconds, the worlds Cloud Video Production ...	Organised - Focused - Vibrant - Awesome!Do you...	What we expect from you:Your key responsibilit...	What you will get from usThrough being part of...	0	1	
2	3	Commissioning Machinery Assistant (CMA)	US, IA, Wever	NaN	NaN	Valor Services provides Workforce Solutions th...	Our client, located in Houston, is actively se...	Implement pre-commissioning and commissioning ...	NaN	0	1	
3	4	Account Executive - Washington DC	US, DC, Washington	Sales	NaN	Our passion for improving quality of life thro...	THE COMPANY: ESRI – Environmental Systems Rese...	EDUCATION: Bachelor's or Master's in GIS, busi...	Our culture is anything but corporate—we have ...	0	1	
4	5	Bill Review Manager	US, FL, Fort Worth	NaN	NaN	SpotSource Solutions LLC is a Global Human Cap...	JOB TITLE: Itemization Review ManagerLOCATION:...	QUALIFICATIONS:RN license in the State of Texa...	Full Benefits Offered	0	1	
...	...	...	...	...	...	...	...	...	...	...	...	
17875	17876	Account Director - Distribution	CA, ON, Toronto	Sales	NaN	Vend is looking for some awesome new talent to...	Just in case this is the first time you've vis...	To ace this role you:Will eat comprehensive St...	What can you expect from us? We have an open cu...	0	1	

Data Pre-Processing:

Below are the statistics of all the numerical features in the dataset.

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
job_id	17880.0	8940.500000	5161.655742	1.0	4470.75	8940.5	13410.25	17880.0
telecommuting	17880.0	0.042897	0.202631	0.0	0.00	0.0	0.00	1.0
has_company_logo	17880.0	0.795302	0.403492	0.0	1.00	1.0	1.00	1.0
has_questions	17880.0	0.491723	0.499945	0.0	0.00	0.0	1.00	1.0
fraudulent	17880.0	0.048434	0.214688	0.0	0.00	0.0	0.00	1.0

We then checked if the dataset has any null values and tried to understand the data type of all the features.

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17880 entries, 0 to 17879
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   job_id               17880 non-null  int64
 1   title                17880 non-null  object
 2   location             17534 non-null  object
 3   department           6333 non-null   object
 4   salary_range         2868 non-null   object
 5   company_profile      14572 non-null  object
 6   description          17879 non-null  object
 7   requirements         15185 non-null  object
 8   benefits             10670 non-null  object
 9   telecommuting        17880 non-null  int64
 10  has_company_logo     17880 non-null  int64
 11  has_questions        17880 non-null  int64
 12  employment_type      14409 non-null  object
 13  required_experience  10830 non-null  object
 14  required_education   9775 non-null   object
 15  industry             12977 non-null  object
 16  function             11425 non-null  object
 17  fraudulent           17880 non-null  int64
dtypes: int64(5), object(13)
memory usage: 2.5+ MB
```

Null values were existing in a couple of columns. No real time data is ever perfectly filled. Below are the snippets of the number of missing values and the percentage of the missing values in each column.

```
[ ] df.isna().mean()*100

job_id                 0.000000
title                  0.000000
location               1.935123
department            64.580537
salary_range          83.959732
company_profile       18.501119
description            0.005593
requirements          15.072707
benefits              40.324385
telecommuting          0.000000
has_company_logo       0.000000
has_questions          0.000000
employment_type       19.412752
required_experience   39.429530
required_education    45.329978
industry              27.421700
function              36.101790
fraudulent             0.000000
dtype: float64
```

```
[ ] df.isna().sum()

job_id                     0
title                      0
location                 346
department             11547
salary_range           15012
company_profile         3308
description                1
requirements            2695
benefits                7210
telecommuting              0
has_company_logo           0
has_questions              0
employment_type         3471
required_experience     7050
required_education      8105
industry                4903
function                6455
fraudulent                 0
dtype: int64
```

In our project, we addressed null values in categorical text columns by substituting them with empty strings (" "). This decision ensures compatibility with subsequent text merging, maintaining data structure and facilitating seamless NLP analysis. The choice of an empty string is neutral and aligns with natural language representation, minimizing potential disruptions. At the end we will be merging these columns to get a single sentence so substituting the nulls with a blank.

```
[ ] df.fillna(' ',inplace=True)
```

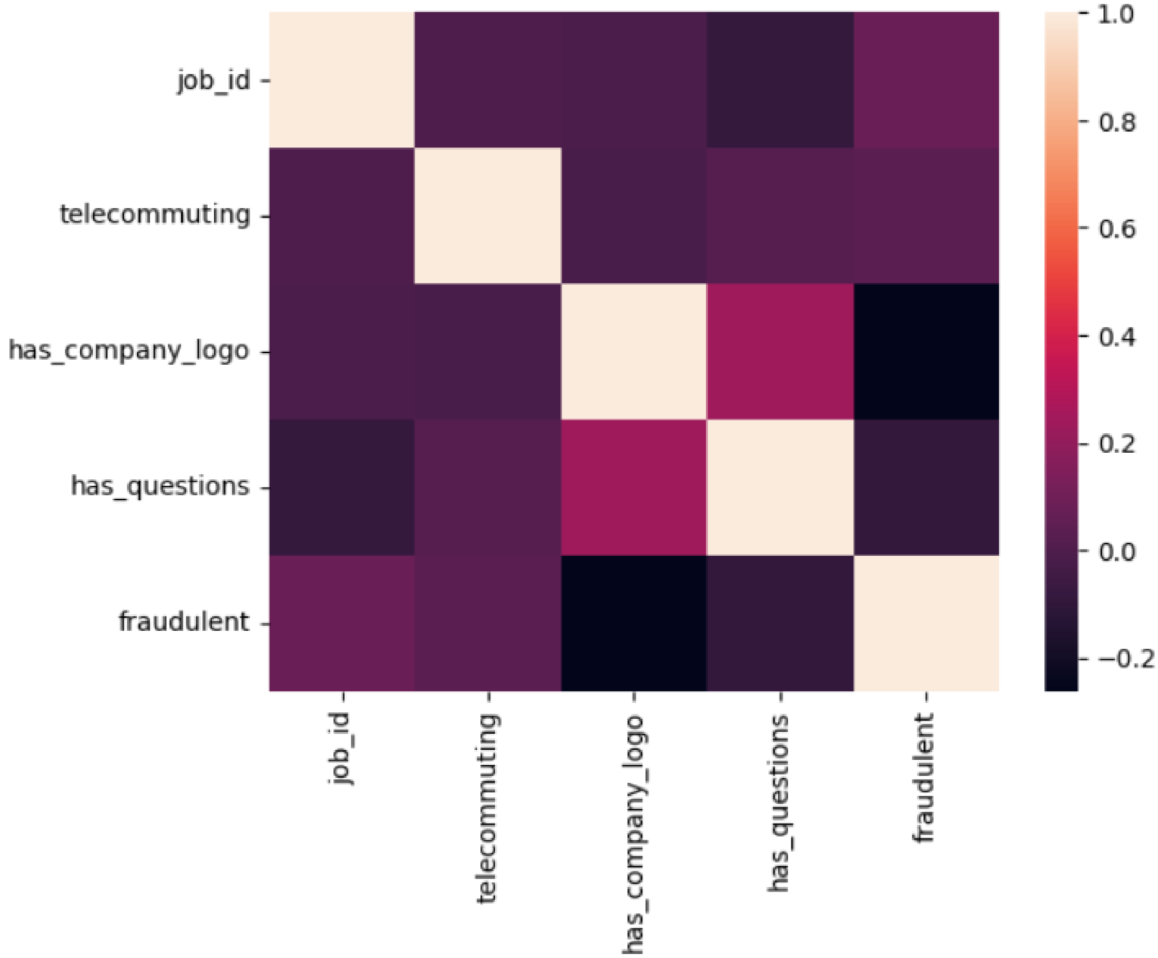
```
df.isna().mean()*100

job_id                 0.0
title                  0.0
location               0.0
department             0.0
salary_range           0.0
company_profile        0.0
description            0.0
requirements           0.0
benefits               0.0
telecommuting          0.0
has_company_logo       0.0
has_questions          0.0
employment_type        0.0
required_experience    0.0
required_education     0.0
industry               0.0
function               0.0
fraudulent             0.0
dtype: float64
```

Exploratory Data Analysis:

We utilized a heatmap to visually represent the correlation matrix of our dataset, providing a concise and intuitive overview of relationships between variables. This graphical representation aids in identifying patterns, dependencies, and potential multicollinearity, facilitating informed feature selection and model-building decisions in our analysis

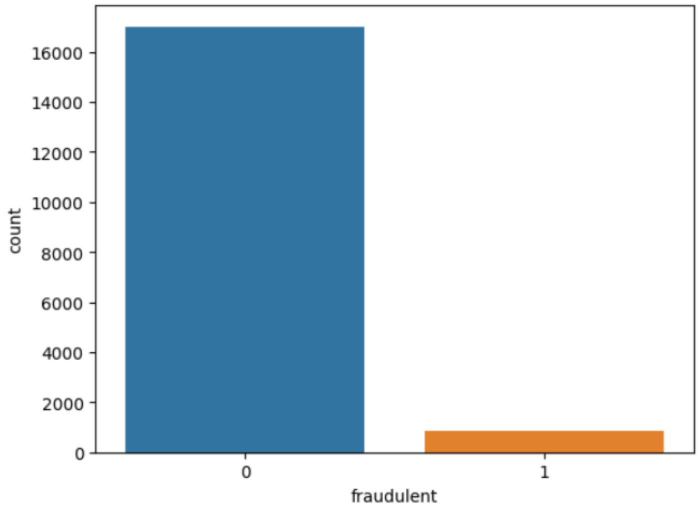
```
sns.heatmap(df.corr());
```

```
<ipython-input-8-f044f2f5ad42>:1: FutureWarning: The default value of numeric_only in Dat
  sns.heatmap(df.corr());
```



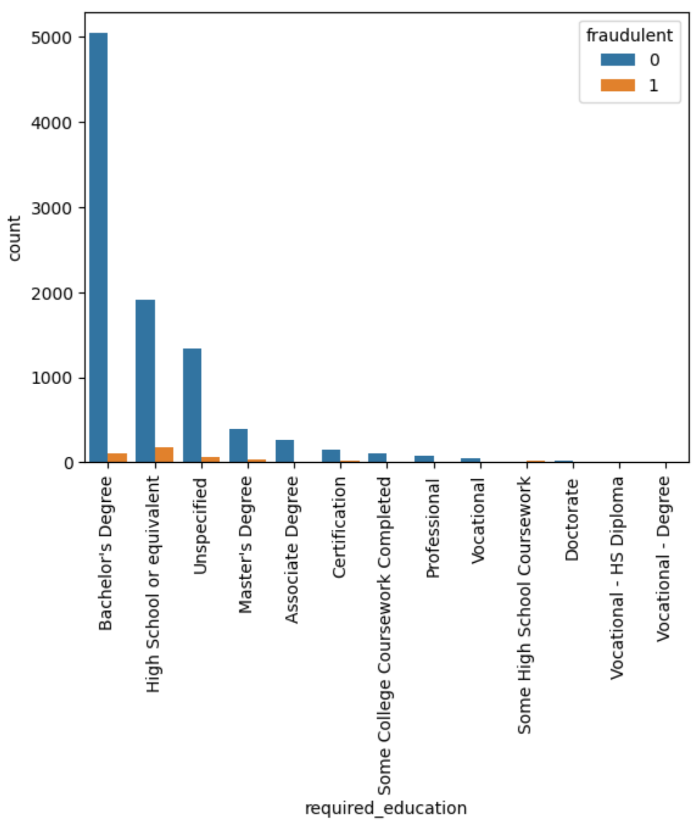
Plot to check the number of Fraudulent and Non-Fraudulent postings

```
[ ] sns.countplot(x='fraudulent',data=df);
```



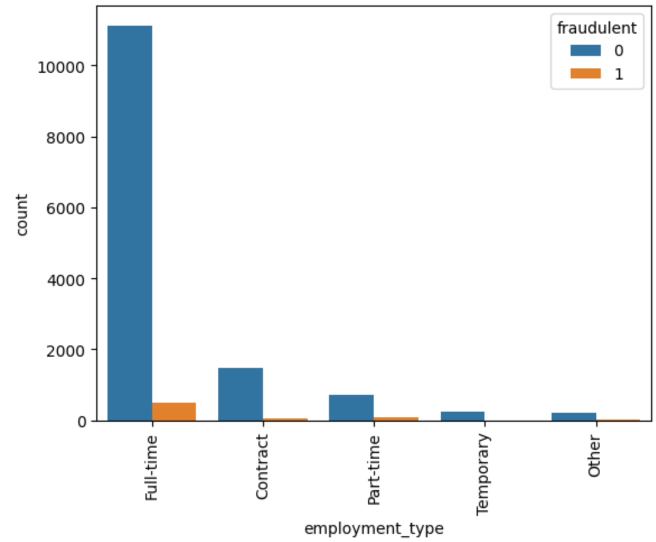
Plot to analyze the number of fraudulent job posting per Educational Category

```
[ ] sns.countplot(x=df['required_education'],hue=df['fraudulent'],order=df['required_education'].value_counts().index)
    plt.xticks(rotation=90);
```



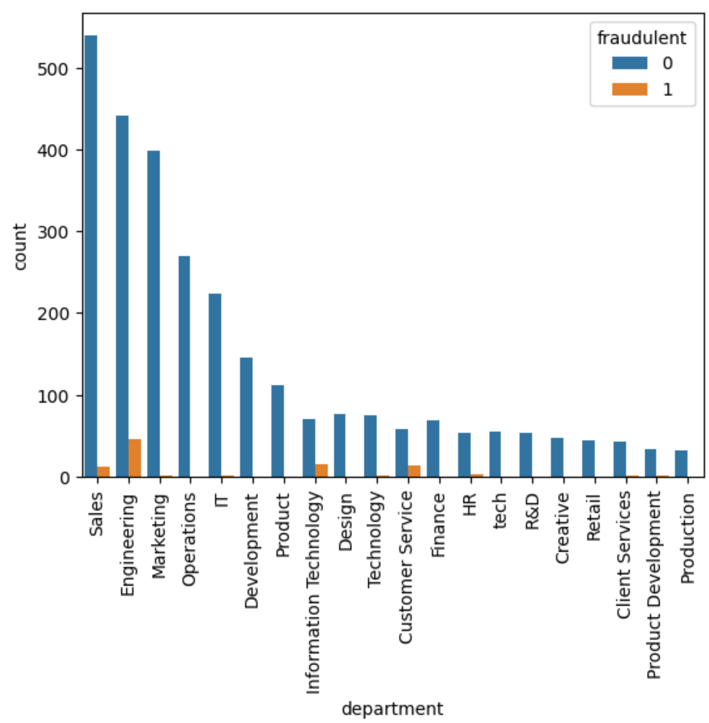
Plot to analyze the number of fraudulent job posting per Employment type.

```
[ ] sns.countplot(x=df['employment_type'],hue=df['fraudulent'],order=df['employment_type'].value_counts().iloc[:10].index)
    plt.xticks(rotation=90);
```



Plot to analyze the number of fraudulent job posting per Department

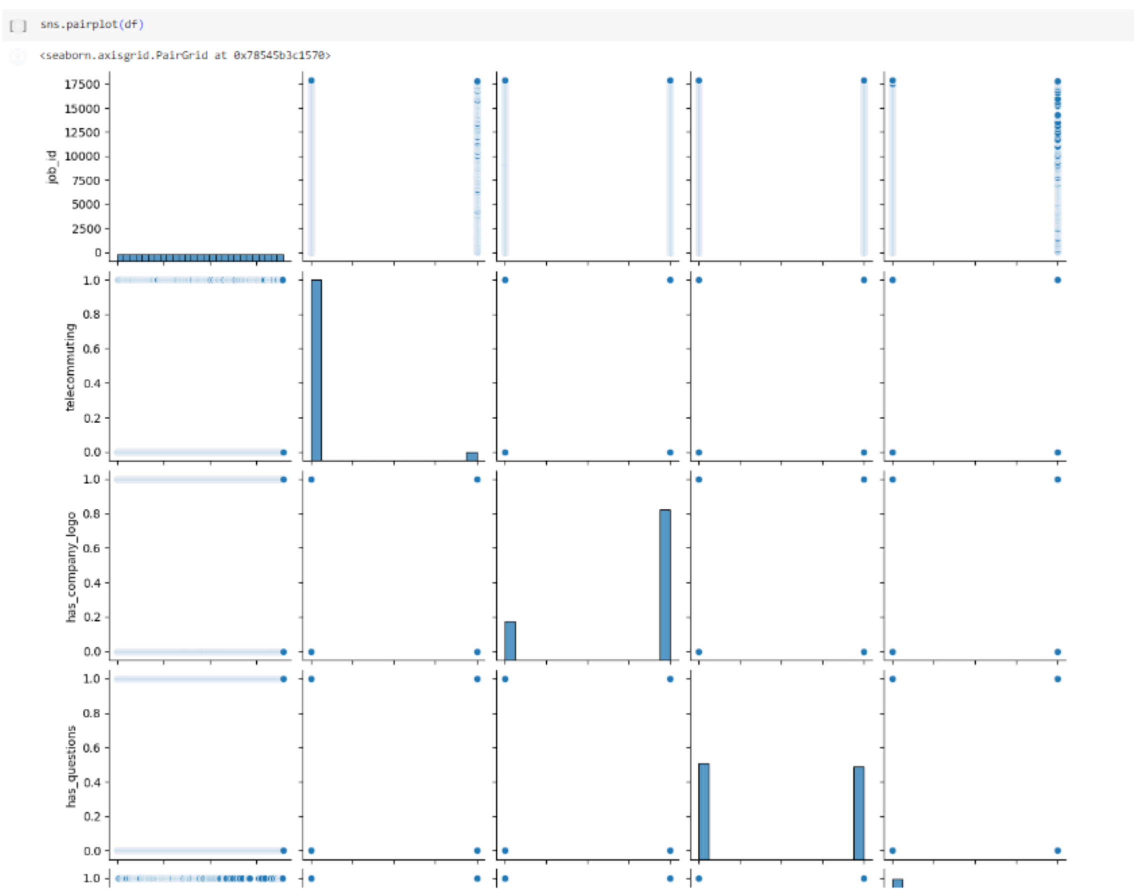
```
sns.countplot(x=df['department'],hue=df['fraudulent'],order=df['department'].value_counts().iloc[:20].index)
plt.xticks(rotation=90);
```



We employed a pair plot to visualize pairwise relationships between key variables in our dataset.

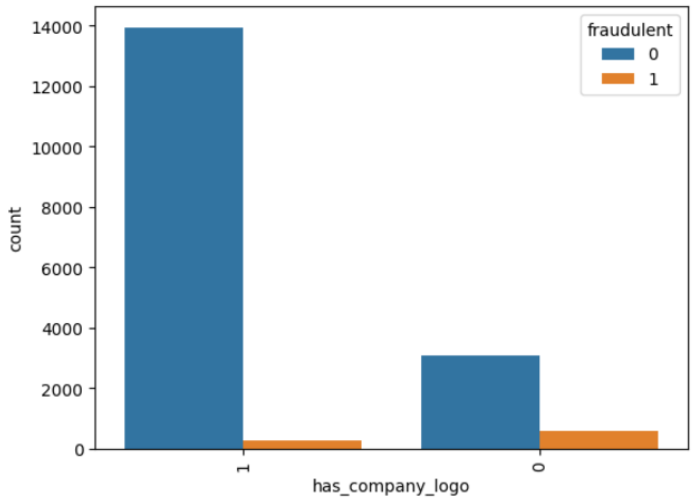
This graphical representation offers quick insights into potential patterns, trends, and outliers, aiding in the identification of meaningful associations.

The pair plot enhances our understanding of variable interactions, informing feature engineering and highlighting potential areas of interest for further analysis.



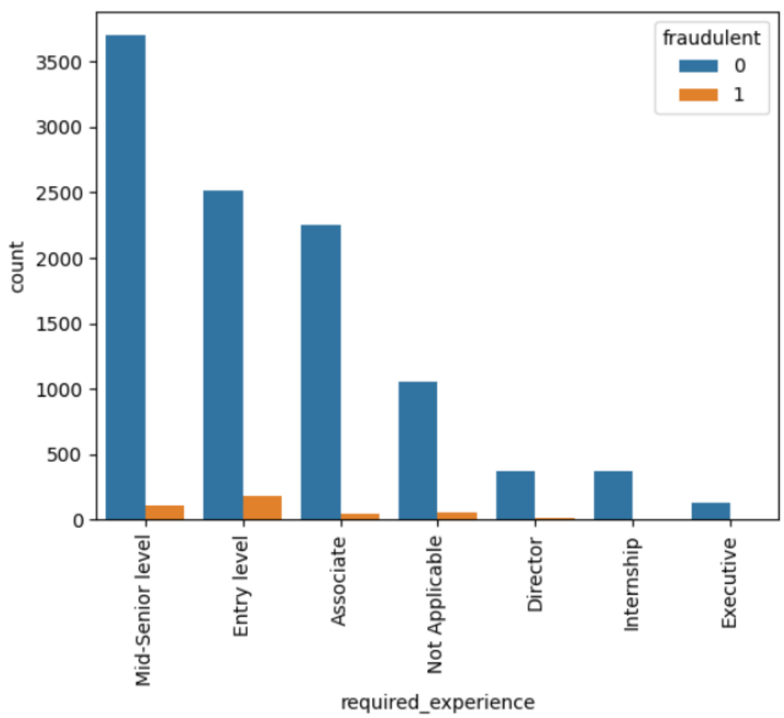
Plot to understand the number of Fraudulent job postings across companies with logo and without logo. From the visualization it is evident that companies without logo have more fraudulent job posting

```
sns.countplot(x=df['has_company_logo'],hue=df['fraudulent'],order=df['has_company_logo'].value_counts().iloc[:10].index)
plt.xticks(rotation=90);
```



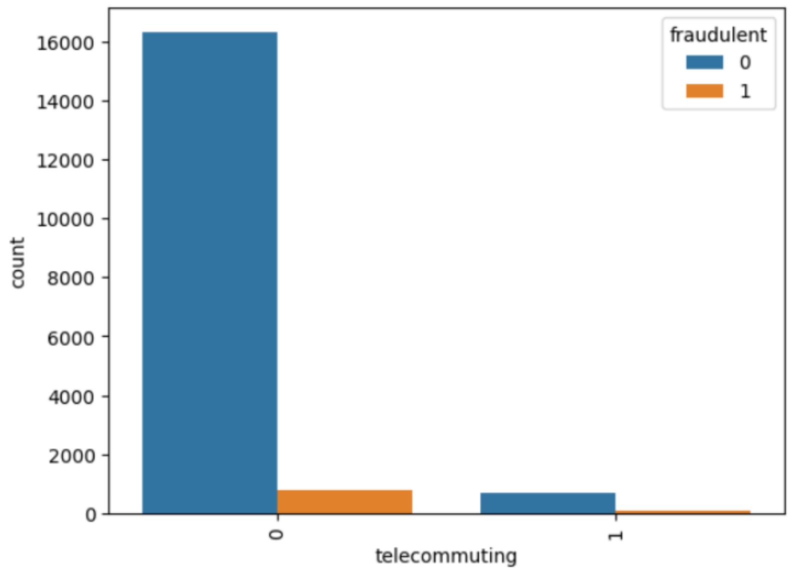
Plot to understand the share of fraudulent job postings across different experience level. We can see that entry level jobs have the most fraudulent postings.

```
sns.countplot(x=df['required_experience'],hue=df['fraudulent'],order=df['required_experience'].value_counts().index)
plt.xticks(rotation=90);
```



Plot to understand the fraudulent job posting in the telecommunicating sector

```
] sns.countplot(x=df['telecommuting'],hue=df['fraudulent'],order=df['telecommuting'].value_counts().iloc[:10].index)
  plt.xticks(rotation=90);
```



Word Cloud

We incorporated various textual columns, such as 'location,' 'department,' 'salary_range,' etc., into a consolidated 'text' column using the below line of code.

```
[ ] df['text']=(df['location']+' '+df['department']+' '+df['salary_range']+' '+df['company_profile']+' '+df['description']+
```

```
[ ] df1=df[['telecommuting','has_company_logo','has_questions','fraudulent','text']]
```

```
df1
```

	telecommuting	has_company_logo	has_questions	fraudulent	text
0	0	1	0	0	US, NY, New York Marketing We're Food52, and...
1	0	1	0	0	NZ, , Auckland Success 90 Seconds, the world...
2	0	1	0	0	US, IA, Wever Valor Services provides Work...
3	0	1	0	0	US, DC, Washington Sales Our passion for imp...
4	0	1	1	0	US, FL, Fort Worth SpotSource Solutions LL...
...	...	...	...	...	...
17875	0	1	1	0	CA, ON, Toronto Sales Vend is looking for so...
17876	0	1	1	0	US, PA, Philadelphia Accounting WebLinc is t...
17877	0	0	0	0	US, TX, Houston We Provide Full Time Perma...
17878	0	0	1	0	NG, LA, Lagos Nemsia Studios is looking ...
17879	0	1	1	0	NZ, N, Wellington Engineering Vend is lookin...

17880 rows × 5 columns

We designed a text cleaning process for a given list. First, we converted the text to lowercase, then removed non-alphanumeric characters, tokenized the text, and eliminate common English stop words. Subsequently, we lemmatized the remaining words using the WordNetLemmatizer. The final step involves calculating the frequency distribution of the cleaned words using NLTK's FreqDist, providing a streamlined approach for preprocessing text data.

```
def clean_review(li):
    a=str(li.tolist()).lower()
    txt=re.sub(r'[^a-z0-9]+',' ',str(a)).strip()
    tokens=word_tokenize(txt)
    stop_words=stopwords.words('english')
    words=[t for t in tokens if t not in stop_words]
    lem=WordNetLemmatizer()
    l=[lem.lemmatize(w) for w in words]
    fd=FreqDist(l)
    return fd
```

```
[ ] wc=WordCloud(background_color='black').generate_from_frequencies(fd)
    plt.imshow(wc)
    plt.axis('off');
```

From the above word cloud, we can see the most frequently appeared terms in a fraudulent job posting.

<u>**Count Vectorizer**</u>:

We employed a CountVectorizer to convert our text data into a numerical format suitable for machine learning models. This technique tokenizes the text and counts the frequency of each word, generating a sparse matrix representation.

```
[ ] cv=CountVectorizer(stop_words='english',max_features=50,ngram_range=(2,2))
    x=cv.fit_transform(df['text'].tolist())
```

```
[ ] cv.get_feature_names_out()
```

```
array(['ability work', 'able work', 'bachelor degree',
       'business development', 'communication skills',
       'competitive salary', 'computer science', 'computer software',
       'customer service', 'degree computer', 'entry level',
       'experience preferred', 'experience working', 'fast growing',
       'fast paced', 'financial services', 'health care', 'high quality',
       'high school', 'ideal candidate', 'information technology',
       'job description', 'join team', 'level bachelor', 'level high',
       'long term', 'mid senior', 'minimum years', 'new york',
       'problem solving', 'project management', 'san francisco',
       'school diploma', 'school equivalent', 'senior level',
       'skills ability', 'social media', 'software development',
       'team members', 'technology services', 'time associate',
       'time entry', 'time mid', 'track record', 'verbal written',
       'work closely', 'work environment', 'written communication',
       'written verbal', 'years experience'], dtype=object)
```

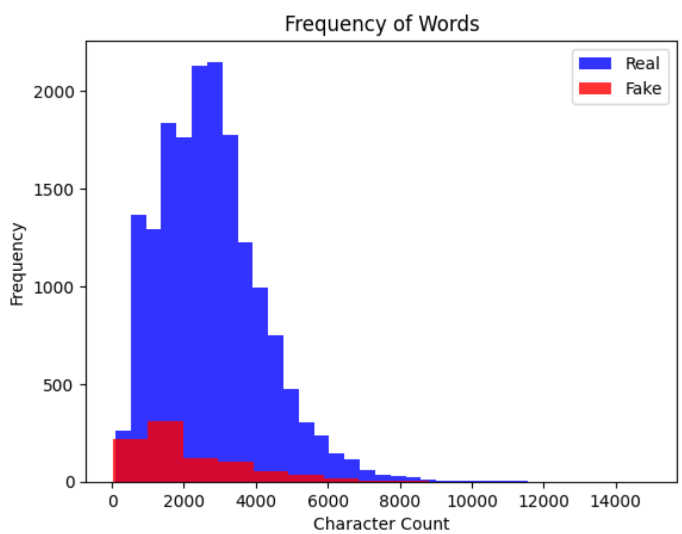
```
[ ] cols=cv.get_feature_names_out()
    dtm=pd.DataFrame(x.toarray(),columns=cols)
    fd2=dtm.sum().to_dict()
    fd2
```

```
{'ability work': 1958,
 'able work': 1301,
 'bachelor degree': 6386,
 'business development': 1387,
 'communication skills': 4042,
 'competitive salary': 1370,
 'computer science': 1365,
 'computer software': 1408,
 'customer service': 6665,
 'degree computer': 1343,
 'entry level': 3274,
 'experience preferred': 1244,
 'experience working': 1672,
 'fast growing': 1870,
 'fast paced': 2817,
 'financial services': 1512,
 'health care': 1251,
 'high quality': 1905,
 'high school': 3002,
 'ideal candidate': 1410,
 'information technology': 4151,
 'job description': 1345,
 'join team': 1396,
 'level bachelor': 2712,
 'level high': 1348,
 'long term': 1961,
 'mid senior': 3841,
```

The word cloud below displays the features that are of most important in our dataset that support real job postings.

```
wc=WordCloud(background_color='black').generate_from_frequencies(fd2)
plt.imshow(wc)
plt.axis('off');
```

The plot below is between the character count and the word count where the x axis represents the length of text in character and the y axis represents the number of words in a text. It allows the comparison between real and fake in terms of the distribution across characters and word count.



Text Vectorization:

TF-IDF, or Term Frequency-Inverse Document Frequency, is a technique in natural language processing. It converts a set of textual documents into numerical vectors, emphasizing the importance of terms based on their frequency in a document and rarity across the entire collection.

We split the data set into train and test and used both TF-IDF and Count Vectorizer.

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

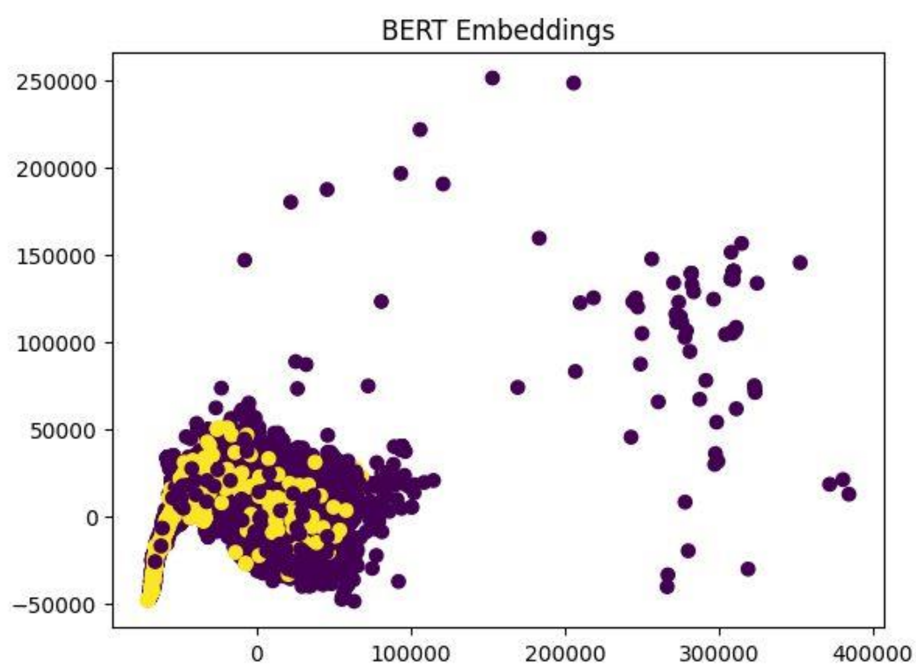
```
[ ] X_train_num = X_train[['telecommuting', 'has_company_logo', 'character_count']]
    X_test_num = X_test[['telecommuting', 'has_company_logo', 'character_count']]
```

```
[ ] count_vectorizer = CountVectorizer(stop_words='english')
    count_train = count_vectorizer.fit_transform(X_train.text.values)
    count_test = count_vectorizer.transform(X_test.text.values)
```

```
[ ] tfidf_vectorizer = TfidfVectorizer(stop_words="english", max_df=1)
    tfidf_train = tfidf_vectorizer.fit_transform(X_train.text)
    tfidf_test = tfidf_vectorizer.transform(X_test.text)
```

Conceptual Embedding Model:

A contextual embedding model (BERT) has been employed to find job Descriptions that are most similar in content. This can help job platforms Detect similar fraudulent listings even if they are not identical in wording



Text Classification:

Multinomial Naïve Bayes:

Text classification using Multinomial Naive Bayes is a probabilistic approach that assigns predefined categories to text documents. The process involves creating a term-document matrix to represent word frequencies in the training data. The algorithm is trained on this matrix, learning the likelihood of each term in specific classes. During prediction, it calculates the probability of a document belonging to each class based on observed term frequencies, ultimately assigning the document to the class with the highest probability.

1. Data Preparation and Feature Extraction:

Convert text data into numerical feature vectors using techniques like CountVectorizer or TF-IDFVectorizer. This step represents the text as a matrix of word counts or TF-IDF values.

2. Training the Model:

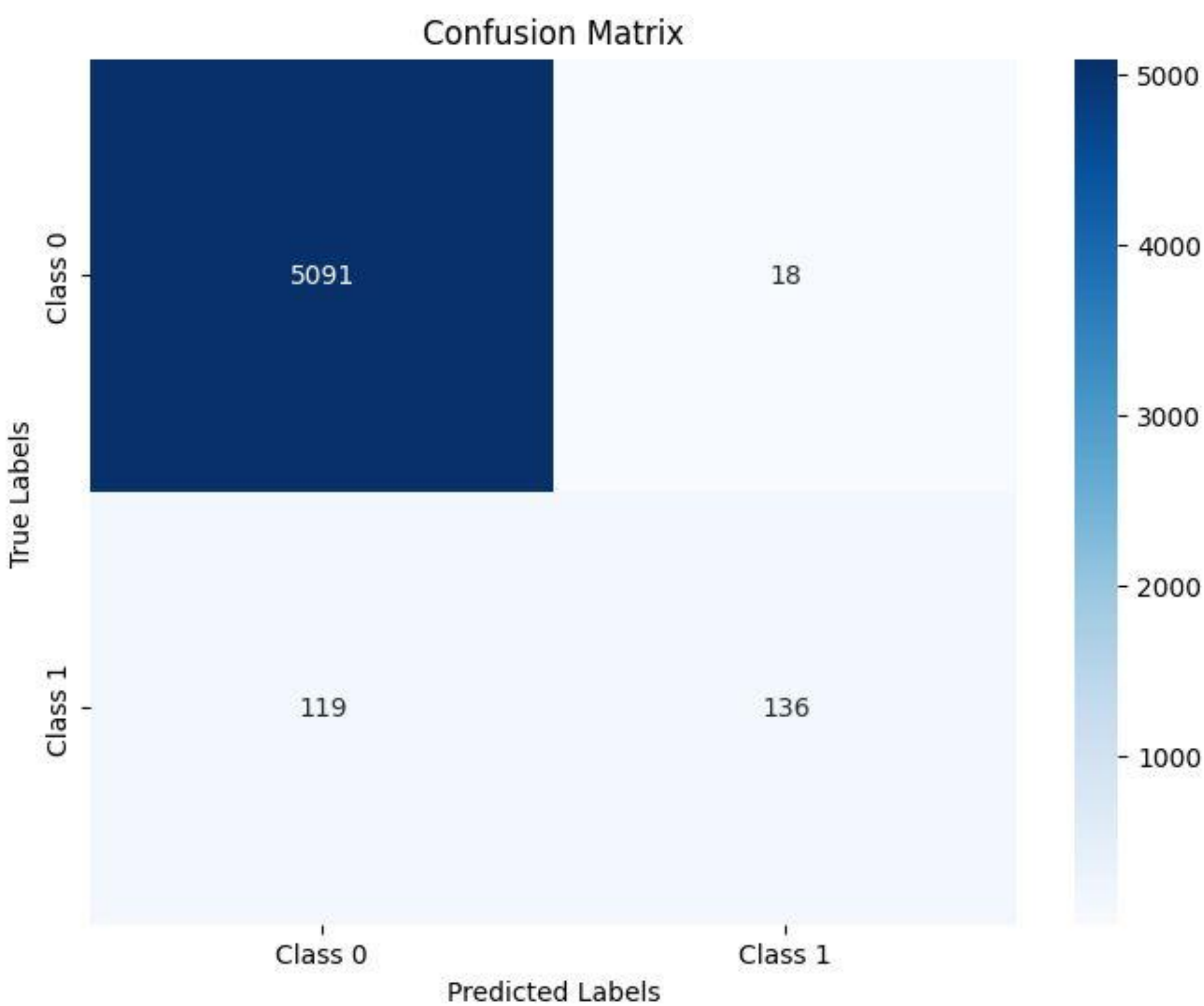
Initialize a MultinomialNB() classifier from scikit-learn's naive_bayes module. Fit the classifier on the training data (features and corresponding labels).

3. Prediction:

Use the trained model to predict labels for new or unseen text data.

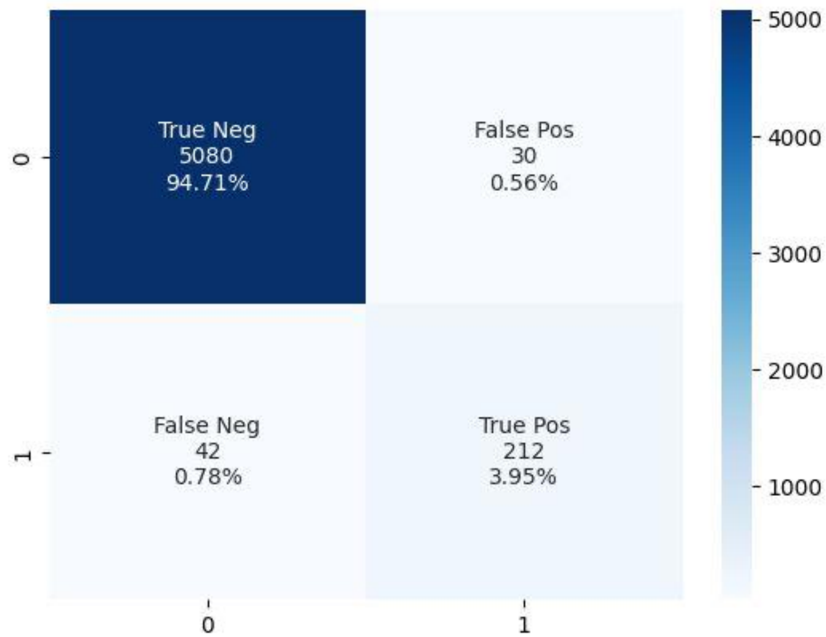
```
[56] nb_classifier = MultinomialNB()
     nb_classifier.fit(count_train, y_train)
     pred = nb_classifier.predict(count_test)
     metrics.accuracy_score(y_test, pred)

     0.9718493661446681
```

```
print(classification_report(y_test,pred))

              precision    recall  f1-score   support

           0       0.98      1.00      0.99      5110
           1       0.88      0.56      0.68       254

    accuracy                           0.98      5364
   macro avg       0.93      0.78      0.83      5364
weighted avg       0.97      0.98      0.97      5364
```



Stochastic Gradient Descent:

Stochastic Gradient Descent (SGD) in text classification is an iterative optimization algorithm that minimizes the loss function by updating model parameters based on small, randomly selected subsets of the training data. SGD adjusts the weights associated with features to minimize the classification error.




```
print(classification_report(y_test, prediction_array))

              precision    recall  f1-score   support

           0       0.99      0.99      0.99      5110
           1       0.88      0.83      0.85       254

    accuracy                           0.99      5364
   macro avg       0.93      0.91      0.92      5364
weighted avg       0.99      0.99      0.99      5364
```

References:

http://emscad.samos.aegean.gr/