Milestone 2 - Report

Project Title : Fake Vs Real Job Posting Analysis

Course - SP20: CMPE-255 Sec 01 - Data Mining

Project Group - **9**

Group Members -
Rutuja Hasurkar (014547793)
Sudarshan Aithal (013638703)
Shivan Desia (010279646)
Jasmine Akkal (013773825)

Table of Contents

1. Introduction / Background

Description :

With ease of access to technologies, the number of jobs and number of job applicants have been significantly increasing in recent years. With this comes fraudulent techniques which misuses the dire situation of applicants by stealing their data for their malicious uses. We are aiming at developing a model to find fake job postings from groups of job postings. The data is provided by The University of the Aegean. This can help a number of job seekers to avoid the fake job postings which can actually expose their personal details in unsafe hands.

Approach :

Main approach of this project is to use data mining techniques to prepare the data for respective models. This is a classification task. So our model will be based on Apriori Association and random forest classification which concentrates on non descriptive attributes like, location, industry, Title, job_id etc. We will be using Data Mining techniques to process the Job Postings data. Using Pearson's correlation and Pearson's Chi Square test to find the dependent variable and feature selection. TF - IDF and pretrained NLP algorithms like ULMFit will be used to develop models to work on descriptive data to find the pattern of words of a fake job posting and test them out on a training set.

Our approach differs from the ones already implemented is on the basis that We will be concentrating on using TF-IDF on selected features of the data, using ULMFit algorithm to analyze the texts, Classification models like Random forest and Apriori association or if time favors, a hybrid model

Related Work :

The previous work on this dataset includes implementation of NLP algorithms using FastAI, BERT analysis and TF-IDF. Also, due to the high imbalance of real job vs fake jobs in the dataset, people have also used various oversampling and undersampling techniques to make the data more balanced for a more accurate prediction.

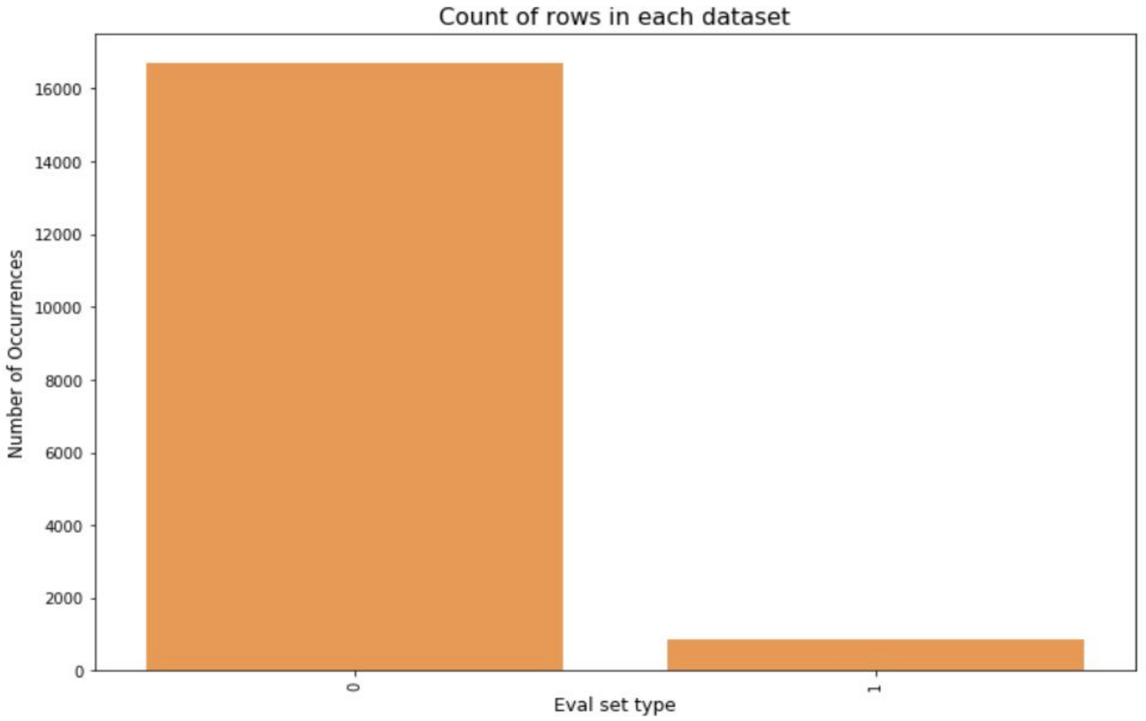
Dataset :

- Dataset is provided by The University of the Aegean.
- It consists of 18 columns.
    - Job_id : Unique Job ID
    - Title : The title of the job ad entry.
    - Location : Geographical location of the job ad.
    - Department : Corporate department (e.g. sales).
    - Salary_range : Indicative salary range (e.g. $50,000-$60,000)
    - Company_profile : A brief company description.
    - Description : The details description of the job ad.
    - Requirements : Enlisted requirements for the job opening.
    - Benefits : Enlisted offered benefits by the employer.
    - Telecommuting : True for telecommuting positions.
    - Has_company_logo : True if company logo is present.
    - Has_questions : True if screening questions are present.
    - Employment_type : Full-type, Part-time, Contract, etc.
    - Required_experience : Executive, Entry level, Intern, etc.
    - Required_education : Doctorate, Master's Degree, Bachelor, etc.
    - Industry : Automotive, IT, Health care, Real estate, etc.
    - Function : Consulting, Engineering, Research, Sales etc.
    - Fraudulent : target - Classification attribute.
- We plan to integrate two datasets to make a superset. The second dataset that we would be using for analysis other than the one mentioned above is fetched from the New York job postings site. This dataset consists of around 3K records and would be used for analysis and implementation of unsupervised models.

2. Pre processing and Exploratory Data Analysis

Data pre processing:

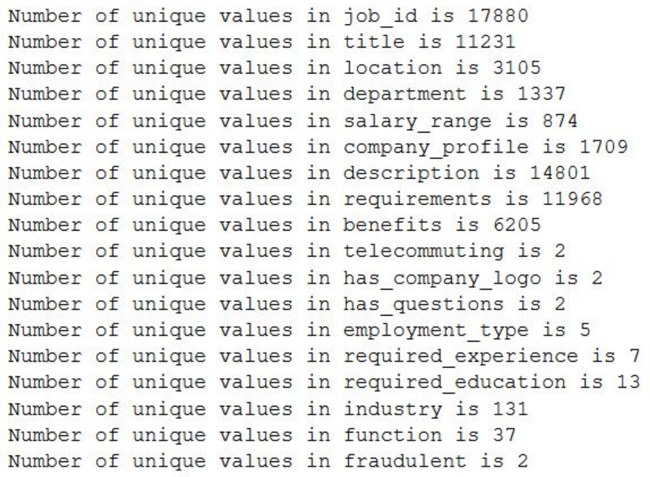
- The dataset was provided by the Laboratory of Information & Communication Systems Security Of The University of the Aegean.
- The dataset consists of 18k job descriptions with 800 fake job descriptions.



- There are 18 columns in the dataset
  - Job_id : Unique Job ID
  - Title : The title of the job ad entry.
  - Location : Geographical location of the job ad.
  - Department : Corporate department (e.g. sales).
  - Salary_range : Indicative salary range (e.g. $50,000-$60,000)
  - Company_profile : A brief company description.
  - Description : The details description of the job ad.
  - Requirements : Enlisted requirements for the job opening.
  - Benefits : Enlisted offered benefits by the employer.
  - Telecommuting : True for telecommuting positions.
  - Has_company_logo : True if company logo is present.

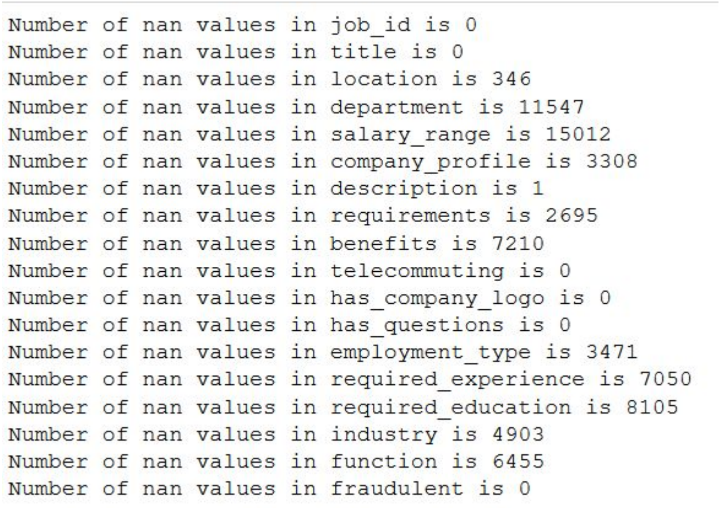
- Has_questions : True if screening questions are present.
- Employment_type : Full-type, Part-time, Contract, etc.
- Required_experience : Executive, Entry level, Intern, etc.
- Required_education : Doctorate, Master’s Degree, Bachelor, etc.
- Industry : Automotive, IT, Health care, Real estate, etc.
- Function : Consulting, Engineering, Research, Sales etc.
- Fraudulent : target - Classification attribute.

- The number of unique values in a column determines the number of states that feature can take. The higher the uniqueness, the more complex the algorithm should be to adapt varying states. Following are the number of unique values each column had.



```
Number of unique values in job_id is 17880
Number of unique values in title is 11231
Number of unique values in location is 3105
Number of unique values in department is 1337
Number of unique values in salary_range is 874
Number of unique values in company_profile is 1709
Number of unique values in description is 14801
Number of unique values in requirements is 11968
Number of unique values in benefits is 6205
Number of unique values in telecommuting is 2
Number of unique values in has_company_logo is 2
Number of unique values in has_questions is 2
Number of unique values in employment_type is 5
Number of unique values in required_experience is 7
Number of unique values in required_education is 13
Number of unique values in industry is 131
Number of unique values in function is 37
Number of unique values in fraudulent is 2
```

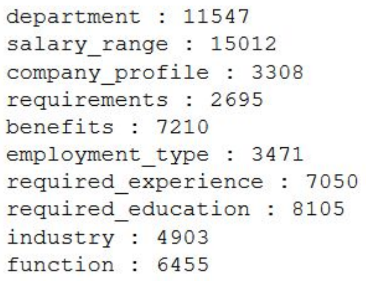
- The multiple columns of the dataset consisted of null values.



```
Number of nan values in job_id is 0
Number of nan values in title is 0
Number of nan values in location is 346
Number of nan values in department is 11547
Number of nan values in salary_range is 15012
Number of nan values in company_profile is 3308
Number of nan values in description is 1
Number of nan values in requirements is 2695
Number of nan values in benefits is 7210
Number of nan values in telecommuting is 0
Number of nan values in has_company_logo is 0
Number of nan values in has_questions is 0
Number of nan values in employment_type is 3471
Number of nan values in required_experience is 7050
Number of nan values in required_education is 8105
Number of nan values in industry is 4903
Number of nan values in function is 6455
Number of nan values in fraudulent is 0
```

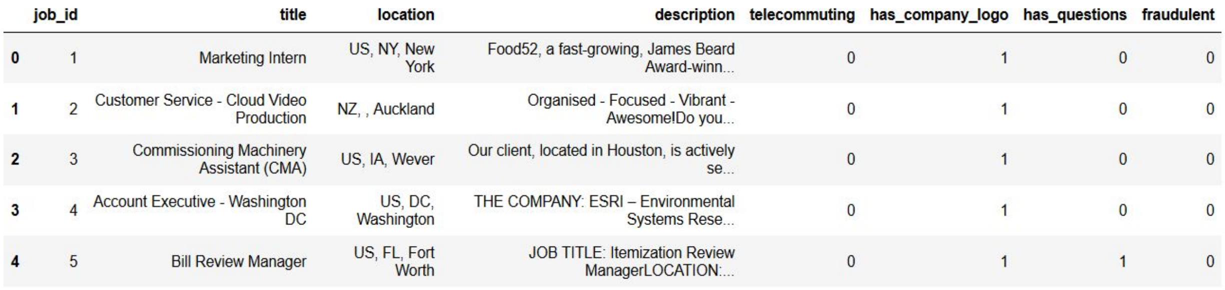
Our task was to eliminate columns with at least 10% of Null value depending on the algorithms' focus. As sparse columns will not contribute much to the learning.

- Following columns had more than 10% of Null values.



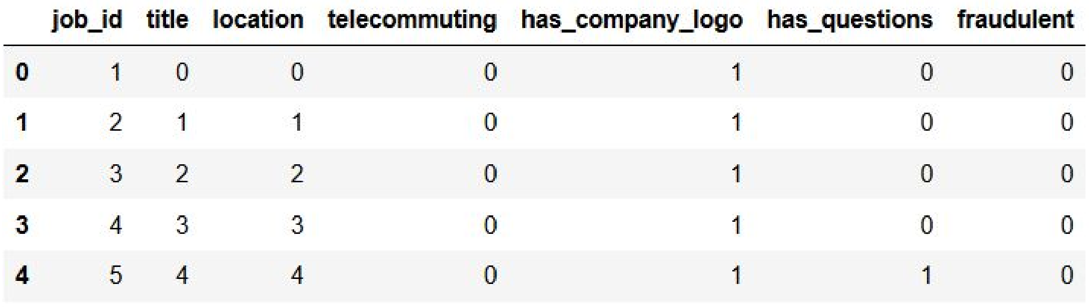
```
department : 11547
salary_range : 15012
company_profile : 3308
requirements : 2695
benefits : 7210
employment_type : 3471
required_experience : 7050
required_education : 8105
industry : 4903
function : 6455
```

- A peek into the dataset after the dataset columns with more than 10% null values were removed.



	job_id	title	location	description	telecommuting	has_company_logo	has_questions	fraudulent
0	1	Marketing Intern	US, NY, New York	Food52, a fast-growing, James Beard Award-winn...	0	1	0	0
1	2	Customer Service - Cloud Video Production	NZ, , Auckland	Organised - Focused - Vibrant - Awesome!Do you...	0	1	0	0
2	3	Commissioning Machinery Assistant (CMA)	US, IA, Wever	Our client, located in Houston, is actively se...	0	1	0	0
3	4	Account Executive - Washington DC	US, DC, Washington	THE COMPANY: ESRI – Environmental Systems Rese...	0	1	0	0
4	5	Bill Review Manager	US, FL, Fort Worth	JOB TITLE: Itemization Review ManagerLOCATION:...	0	1	1	0

- We used tokenization to convert non-numerical attributes to numerical attributes that can be further used by the algorithm to classify. For Non NLP classifier algorithms, using descriptive information for learning may not be an easy task as to take tokenizing Descriptive information may lead to all unique numbers and learning can be hard.

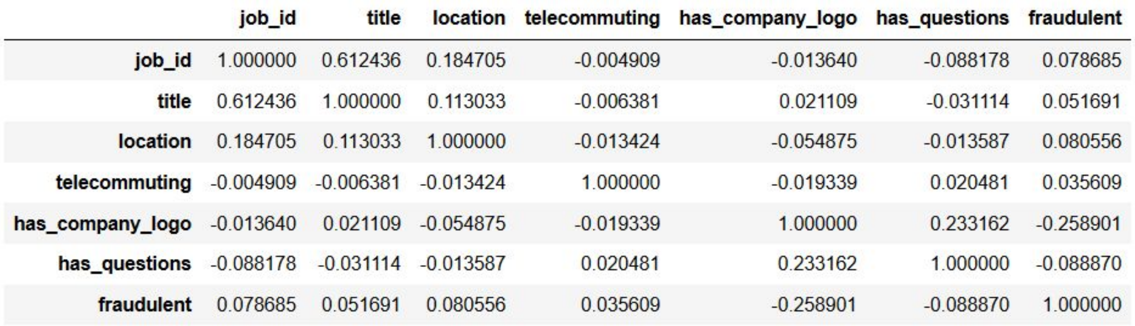


	job_id	title	location	telecommuting	has_company_logo	has_questions	fraudulent
0	1	0	0	0	1	0	0
1	2	1	1	0	1	0	0
2	3	2	2	0	1	0	0
3	4	3	3	0	1	0	0
4	5	4	4	0	1	1	0

Depending on the algorithm, preprocessing of the data can be varied to yield efficient learning

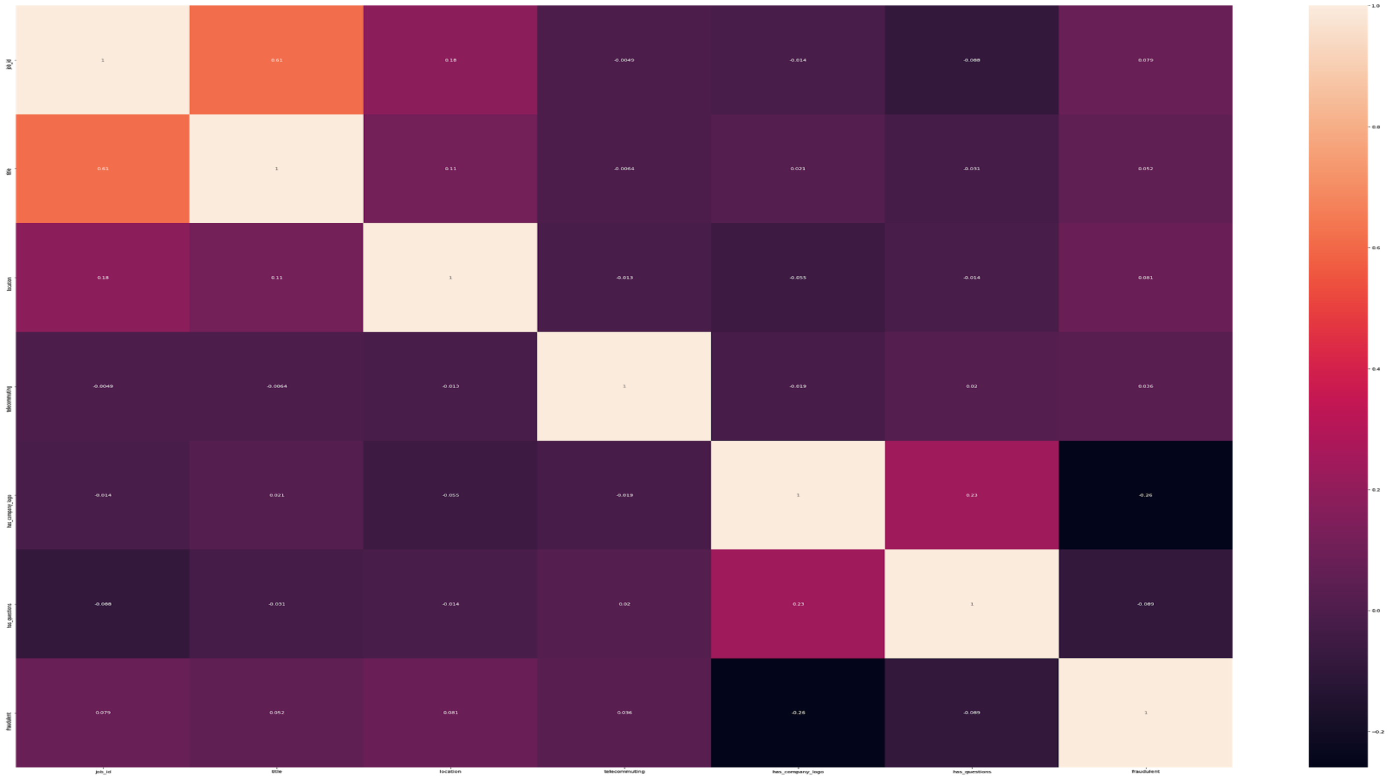
- To determine dependent attributes that contribute to the final prediction of a fake or real job, we are using Pearson’s Correlation and Pearson’s Chi Square Test.
    - Pearson’s correlation is used to determine a coefficient between two attributes which defines if those two attributes are linearly dependent or not. If the
      Coefficient value is 0, Then the two variables are independent of each other
      Coefficient value is 1, the two variables are dependent of each other
    - Pearson’s Chi Square Test is also used to define linear dependency between two attributes. By defining a null hypothesis, we determine the observed frequency of an attribute by using another attribute and compare it with the original value. We get a “*p value*” that determines if the hypothesis chosen to correlate two attributes should be accepted or rejected.
    - A good practice is to select p-value and pearson’s correlation coefficient to be at least > 0.05

- Pearson’s correlation of factorized, description removed, greater than 10% null valued column removed data set is:

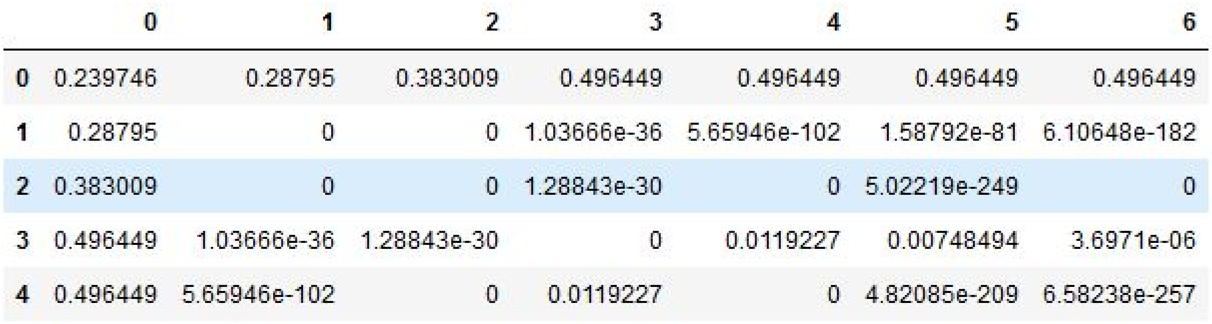


	job_id	title	location	telecommuting	has_company_logo	has_questions	fraudulent
job_id	1.000000	0.612436	0.184705	-0.004909	-0.013640	-0.088178	0.078685
title	0.612436	1.000000	0.113033	-0.006381	0.021109	-0.031114	0.051691
location	0.184705	0.113033	1.000000	-0.013424	-0.054875	-0.013587	0.080556
telecommuting	-0.004909	-0.006381	-0.013424	1.000000	-0.019339	0.020481	0.035609
has_company_logo	-0.013640	0.021109	-0.054875	-0.019339	1.000000	0.233162	-0.258901
has_questions	-0.088178	-0.031114	-0.013587	0.020481	0.233162	1.000000	-0.088870
fraudulent	0.078685	0.051691	0.080556	0.035609	-0.258901	-0.088870	1.000000

- Heatmap of Pearson’s correlation.

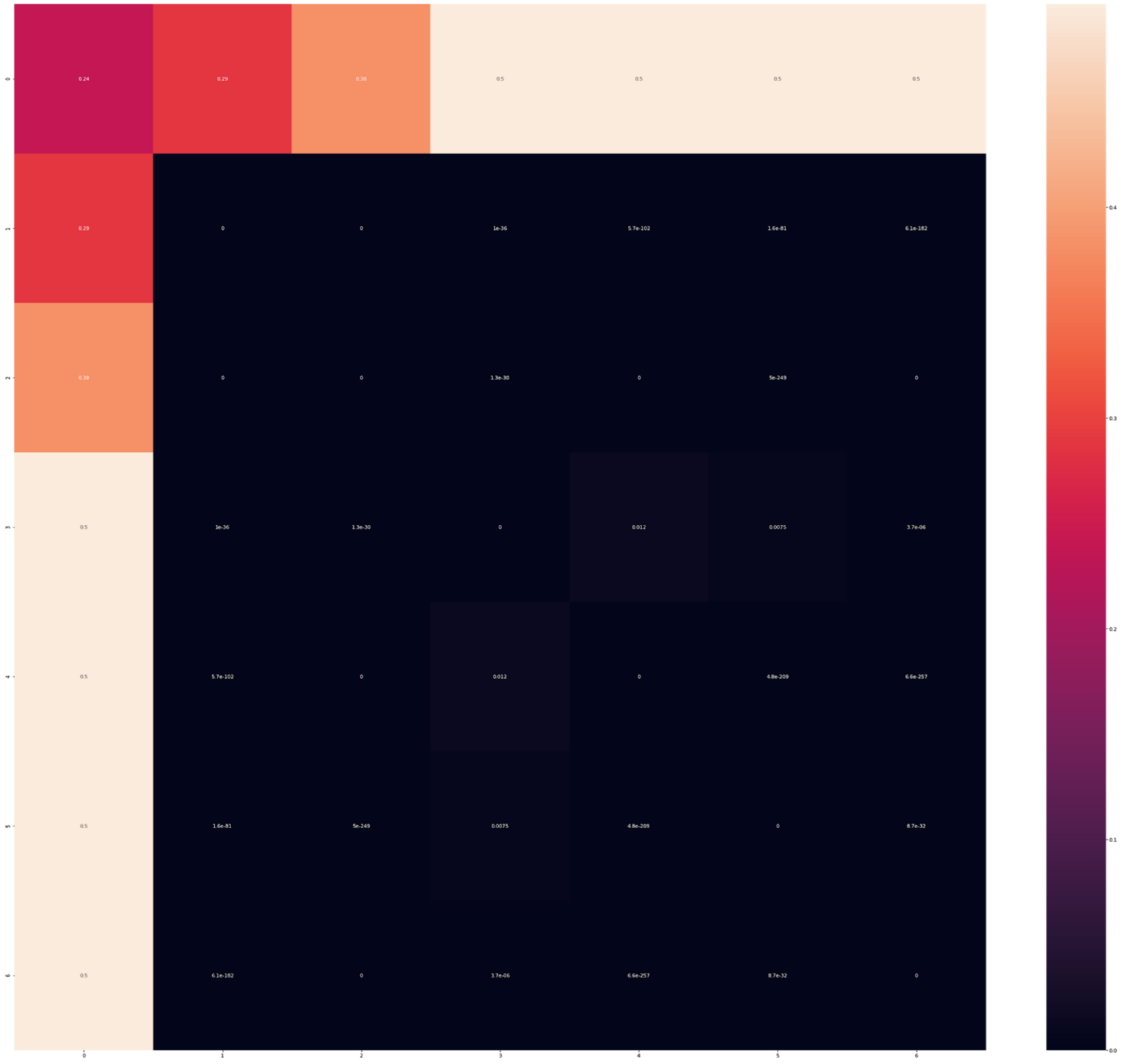


- P values for factorized, description removed, greater than 10% null valued column removed data set is:



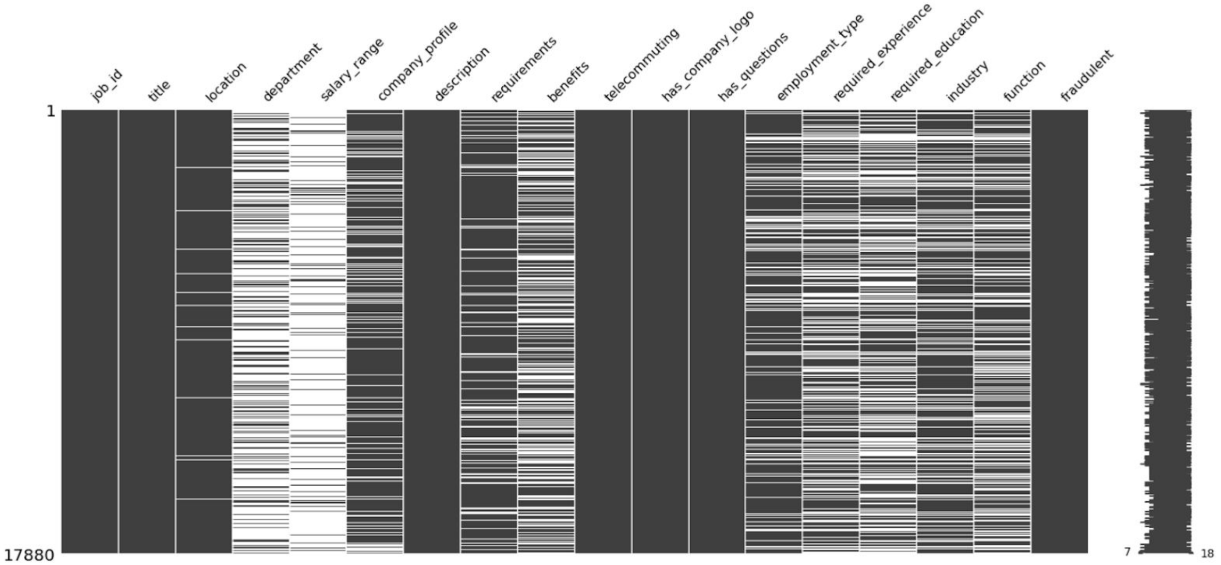
	0	1	2	3	4	5	6
0	0.239746	0.28795	0.383009	0.496449	0.496449	0.496449	0.496449
1	0.28795	0	0	1.03666e-36	5.65946e-102	1.58792e-81	6.10648e-182
2	0.383009	0	0	1.28843e-30	0	5.02219e-249	0
3	0.496449	1.03666e-36	1.28843e-30	0	0.0119227	0.00748494	3.6971e-06
4	0.496449	5.65946e-102	0	0.0119227	0	4.82085e-209	6.58238e-257

- Heatmap for P values.

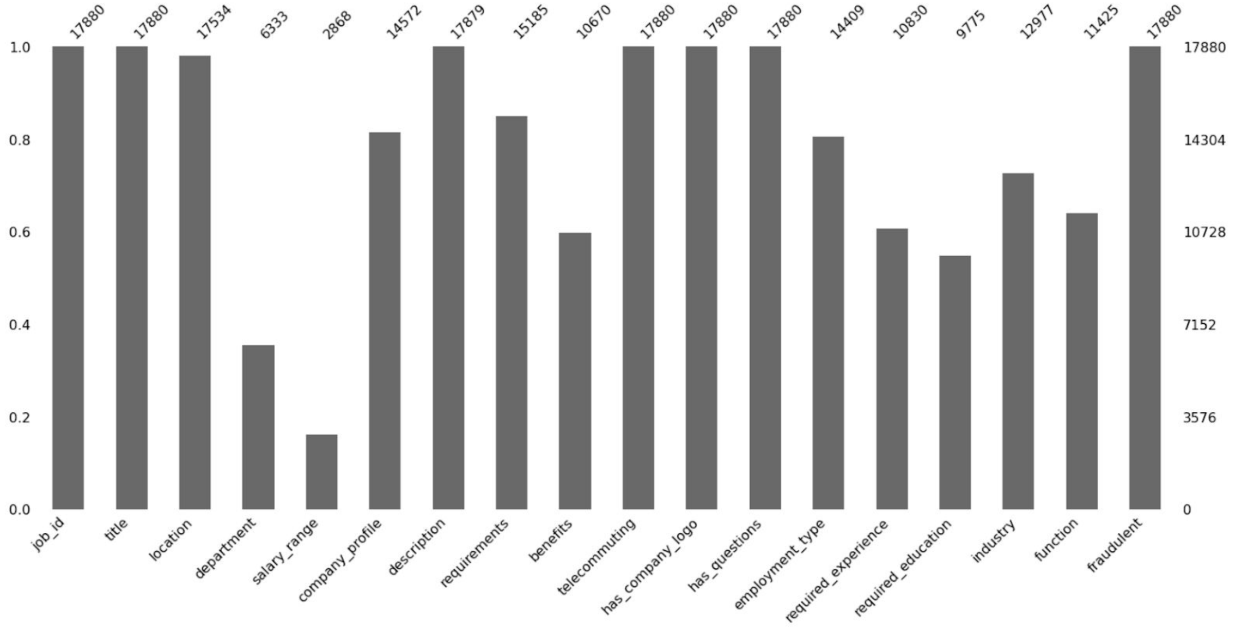


Exploratory Data Analysis (EDA):

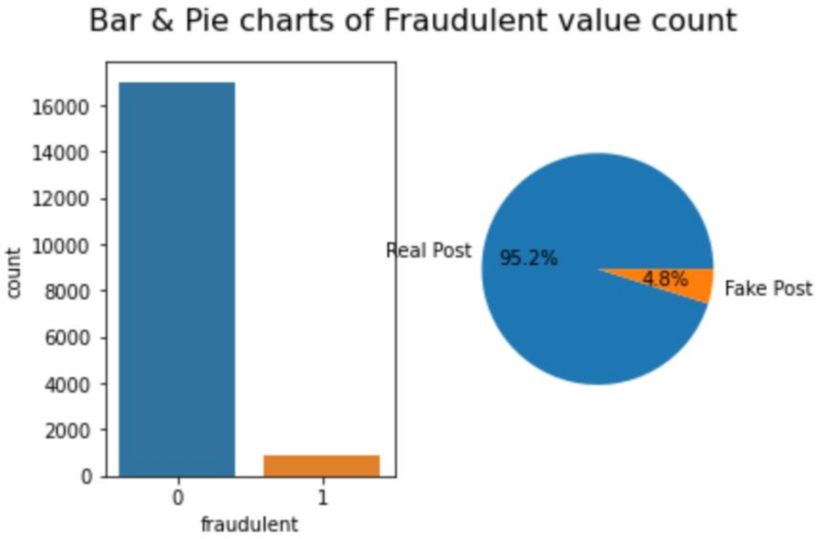
- Visualizing the missing values in the dataset. We use missingno library to plot and get the visual representation of the records and columns representing the cells which have missing values. In the following plot the black are filled records whereas the white spaces represent the missing field in the dataset.



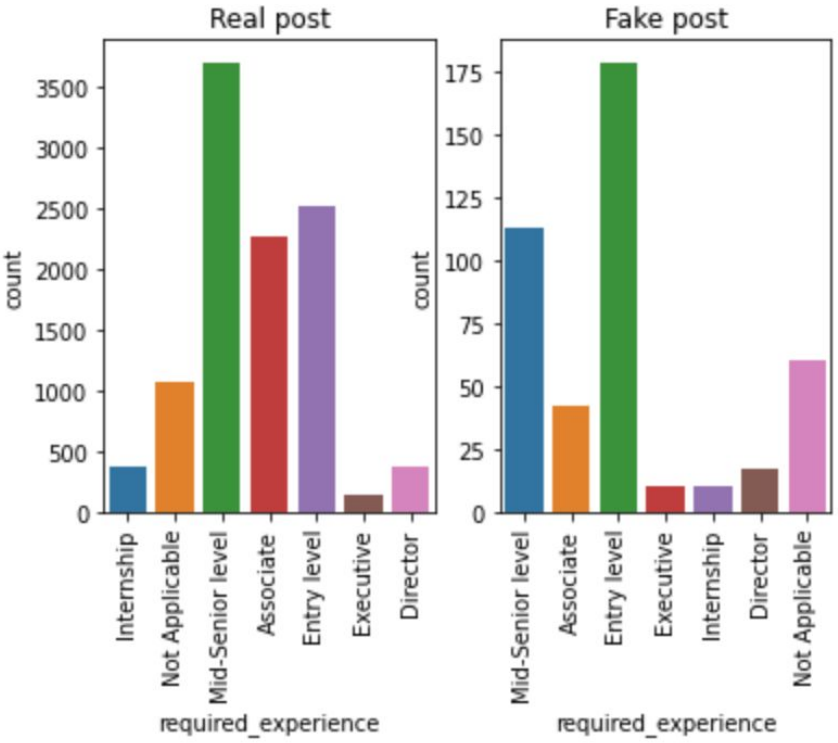
- Plotting a bar graph for representing the missing value count in the dataset.



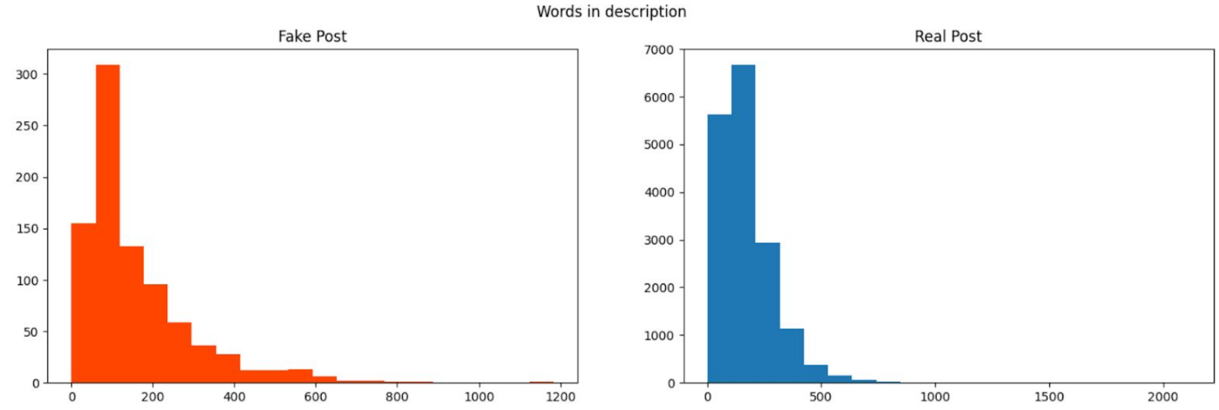
- Graphical representation of the count value of the target field. For our dataset we have the column “fraudulent” as our target field. We use a bar chart and pie chart to display the count of this column.



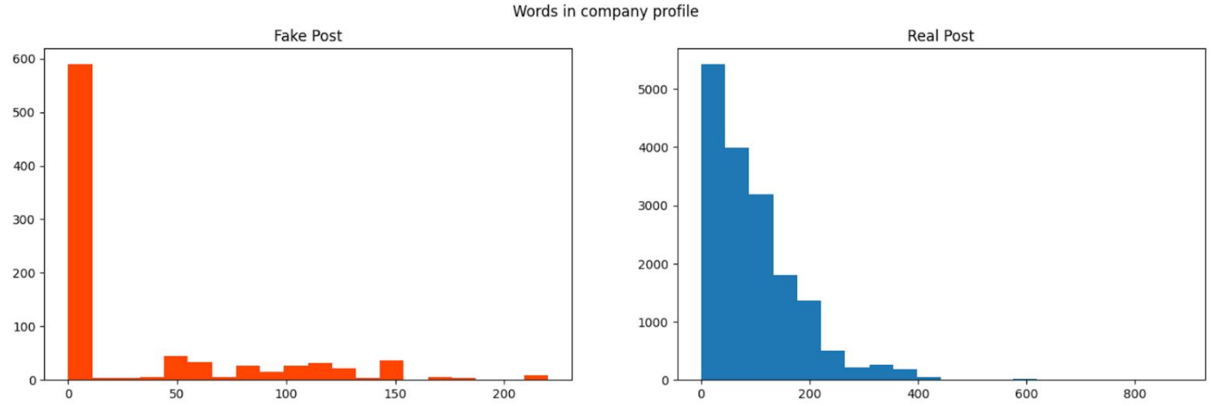
- Analysing the column of the dataset “requird_experience”. We are doing the following analysis on the training dataset.



- The job description plays a major role in understanding if the job post is real or fake. So we analyse the column “description” of the dataset to count the number of words. We observe how the trend is with respect to the word count and real / fake job posting.



- Word count analysis on the column “company_profile”.



3. Method Description:

3.1) Random Forest Classifier

Random Forest Classifier is an ensemble learning algorithm. Ensembled algorithms can be explained as those which implements a combination of more than a single algorithm of similar or different types for classification purpose. Random FOrest Classifier can be explained to be an algorithm which creates a set of decision trees by selection of random subsets of the training dataset. It then accumulates the results from different smaller decision trees to decide the final class of the test object.

Suppose we have a training dataset as : [A1, A2, A3, A4, A5] which are labelled as [L1, L2, L3, L4, L5]. The random forest classifier divides the training set into smaller subsets like [A1, A2, A3], [A2, A4, A5], [A1, A3, A5], [A3, A4, A5] and likewise. The algorithm calculates the accuracy for individual subsets and then aggregates the results of the subsets to finally classify the data from the test dataset. [7.2]

Random forest classifier models are used over decision trees because a single decision tree may be prone to noise. But, with results from multiple decision trees aggregated may reduce the effect of the noise, thus giving more accurate results and efficiency on large data bases. The parameters used by the random forest classifier are the total number of trees to be generated and decision tree related parameters like minimum split criteria.

The reason behind selecting the random forest classifier model for the job posting analysis and fake job detection dataset is that this dataset has around 18K records over 18 columns, which comprises missing data. Random forest classifier model is well known for being able to handle large input variables, it gives estimates of what variables are important in the classification. Further, it has an effective method for estimating the missing data and maintains accuracy when a large proportion of the data are missing.

3.2) TF-IDF

TF-IDF is a numeric statistic that helps determine how important a word is for a document in a document or corpus. It has two parts, namely term frequency and inverse document frequency. A lot of times, these parts might be used seperately in various situations. However, TF-IDF is the combination of these two statistics.

Term Frequency: To find the term frequency of a phrase or a query, we first eliminate all the documents that do not contain the word. This usually eliminates many documents. Following this, we count the number of times each term of the phrase occurs in each document that contains the words. This process basically helps us determine the ‘weight’ of each word in the phrase.

Inverse document frequency: Term frequency incorrectly gives a lot of weight to words that occur more frequently, but might be very less significant in the actual analysis, and gives less importance to meaningful terms that might have more significance. Such extremely frequent words (called ‘stop words’ eg: ‘the’) should first be removed before we apply TF-IDF. However, to diminish the weight of other words that might occur very frequently and increase the weights of those words that occur rarely, an inverse document frequency is incorporated. Inverse document frequency is a measure of how much information a word provides for the analysis.
To calculate the inverse document frequency, we first divide the total number of documents by the number of documents that contain the term, followed by calculating the logarithm of the value we obtain.

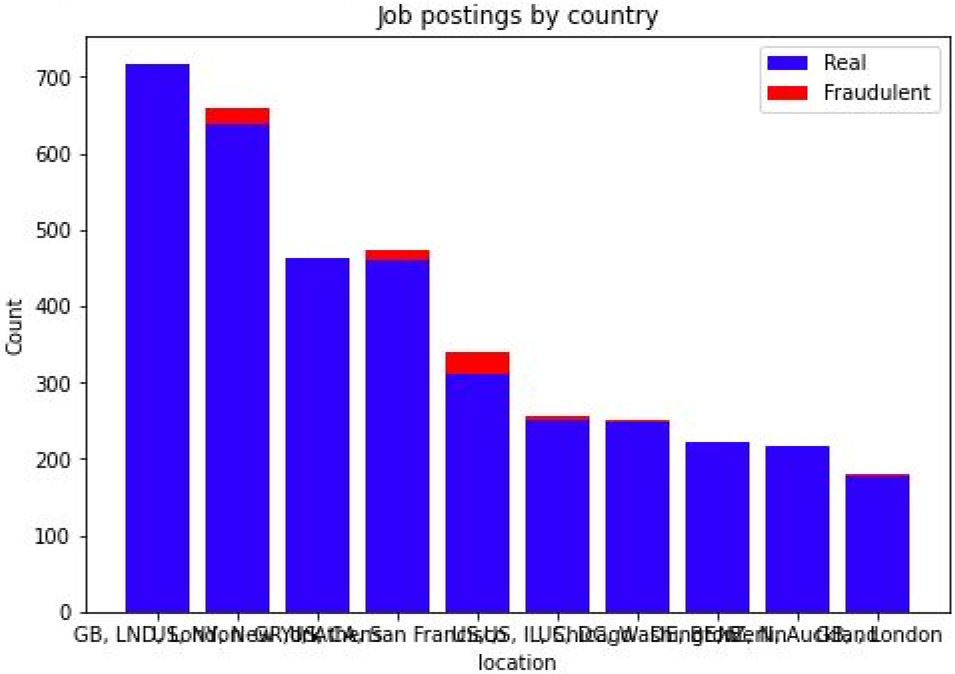
TF-IDF is the product of TF and IDF.

3.3) Natural-Language Programming Countvectorizer and ULMFit:

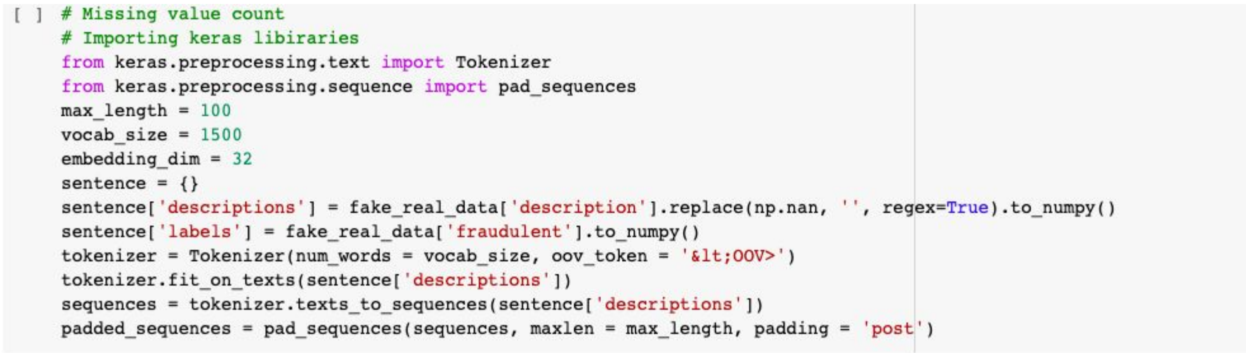
Count vectorizer Model : Predictive Modeling requires text data to be specially prepared before we can start working on it. The process of tokenization involves removing words from the text and is required to be carried out. Then the words need to be encoded as integers or floating point values for use as input to a machine learning algorithm, called feature extraction (or vectorization).The model is simple in that it throws away all of the order information in the words and focuses on the occurrence of words in a document. This can be done by assigning each word a unique number. Then any document we see can be encoded as a fixed-length vector with the length of the vocabulary of known words. The value in each position in the vector could be filled with a count or frequency of each word in the encoded document. In our code we have already derived accuracy for 5 folds for the count vectorizer model.

ULMFiT: The technique involves training a language model on a large corpus, fine-tuning it for a different and smaller corpus, and then adding a classifier to the end. Our method is based on Universal Language Model Fine-Tuning (ULMFiT). For more context, we invite you to check out the previous blog post that explains it in depth. MultiFiT extends ULMFiT to make it more efficient and more suitable for language modelling beyond English: It utilizes tokenization based on subwords rather than words and employs a QRNN rather than an LSTM. In addition, it leverages a number of other improvements. Subword tokenization ULMFiT uses word-based tokenization, which works well for the morphologically poor English, but results in very large and sparse vocabularies for morphologically rich languages, such as Polish and Turkish. Some languages such as Chinese don't really even have the concept of a "word", so require heuristic segmentation approaches, which tend to be complicated, slow, and inaccurate. On the other extreme as can be seen below, character-based models use individual characters as tokens. While in this case the vocabulary (and thus the number of parameters) can be small, such models require modelling longer dependencies and can thus be harder to train and less expressive than word-based models.

Data Visualization: In my initial approach we tried to clean the data as much as possible to retrieve most useful data for my model to work properly. Below are the figures that depict the visualization and cleaning carried out to understand the entire dataset properly. Below graphs signifies how different attributes are dependent on each other which can help us derive useful insights.

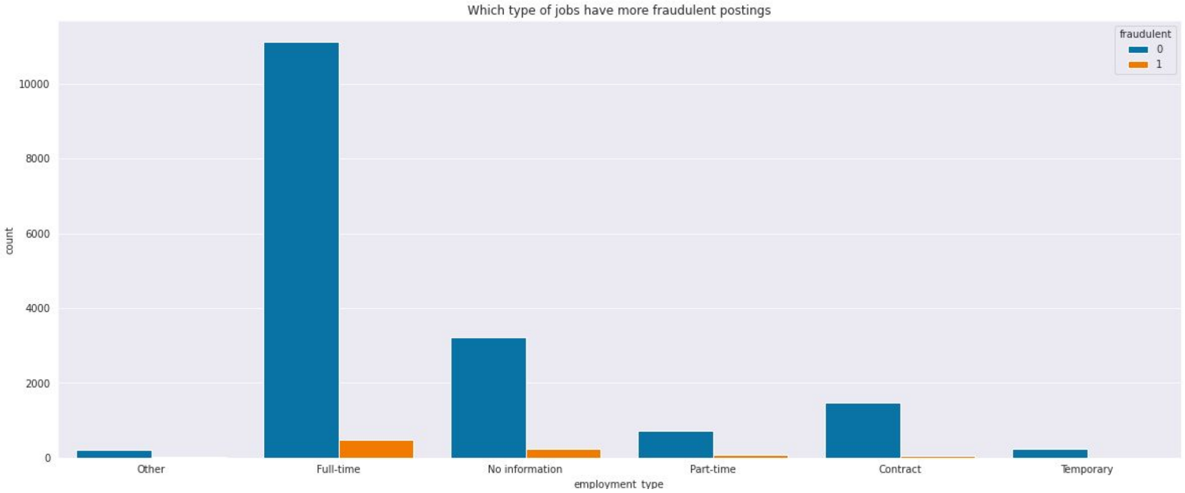


Visualisation of the missing value in the entire dataset

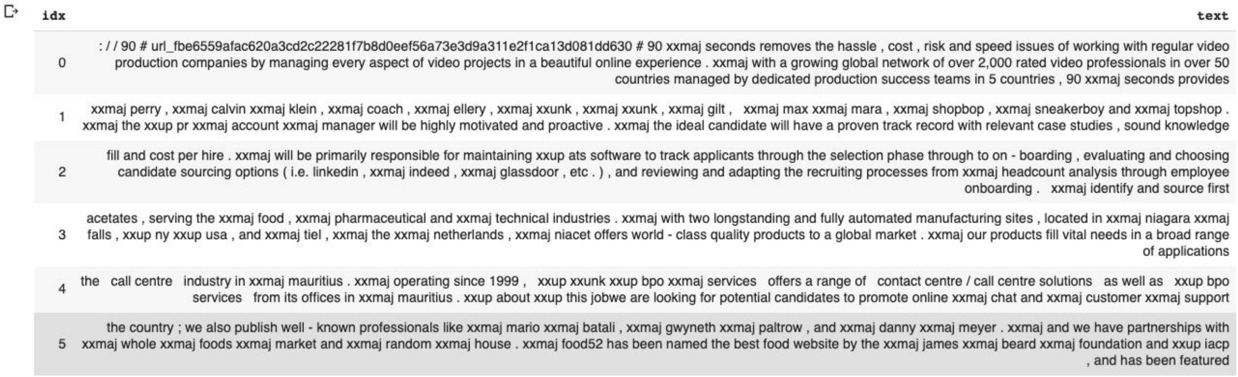


```
[ ]  # Missing value count
     # Importing keras libiraries
     from keras.preprocessing.text import Tokenizer
     from keras.preprocessing.sequence import pad_sequences
     max_length = 100
     vocab_size = 1500
     embedding_dim = 32
     sentence = {}
     sentence['descriptions'] = fake_real_data['description'].replace(np.nan, '', regex=True).to_numpy()
     sentence['labels'] = fake_real_data['fraudulent'].to_numpy()
     tokenizer = Tokenizer(num_words = vocab_size, oov_token = '&lt;OOV>')
     tokenizer.fit_on_texts(sentence['descriptions'])
     sequences = tokenizer.texts_to_sequences(sentence['descriptions'])
     padded_sequences = pad_sequences(sequences, maxlen = max_length, padding = 'post')
```

Missing value Count

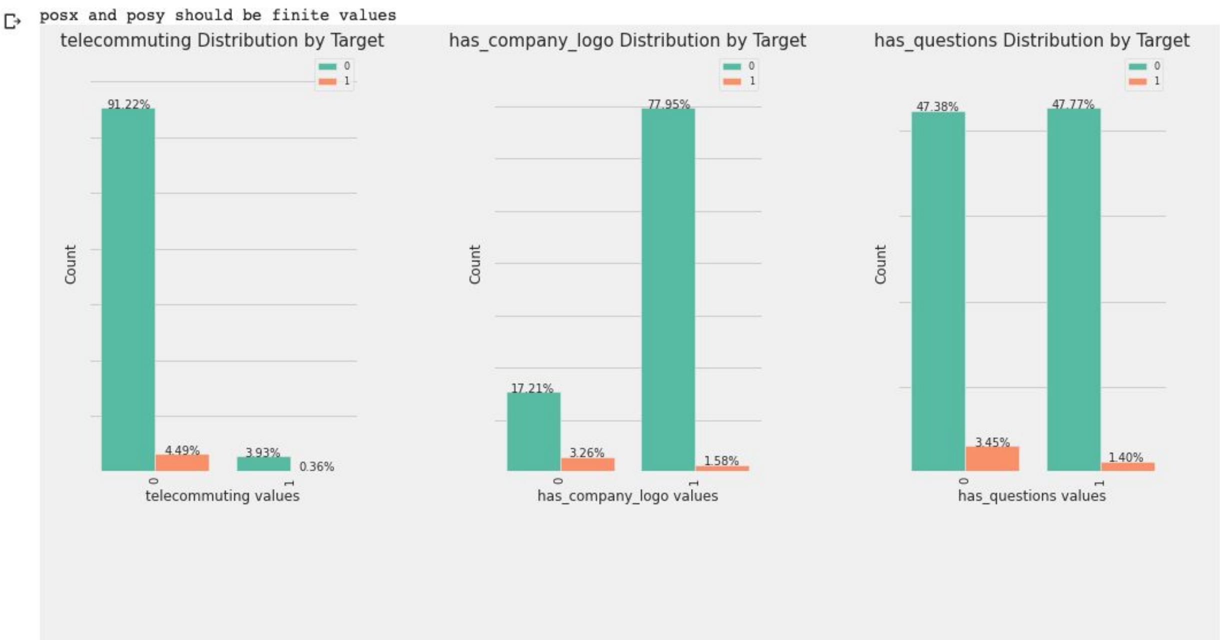


Employment type vs count visualization

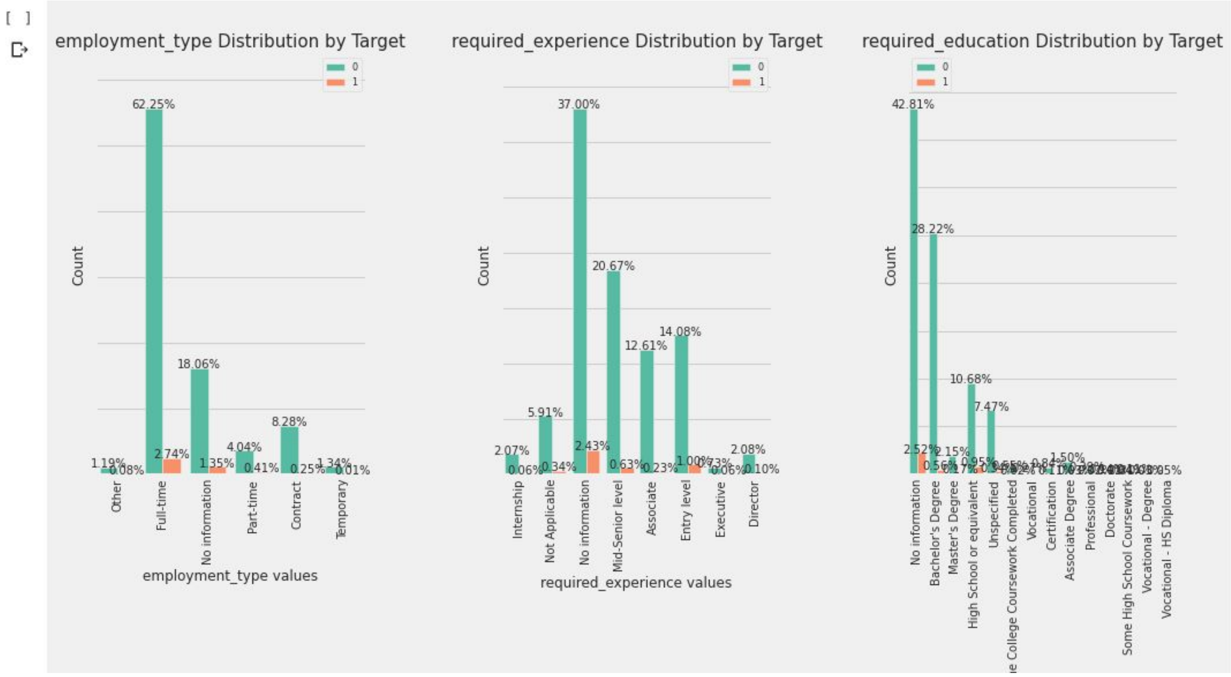


idx	text
0	: / / 90 # url_fbe6559afac620a3cd2c22281f7b8d0eef56a73e3d9a311e2f1ca13d081dd630 # 90 xxmaj seconds removes the hassle , cost , risk and speed issues of working with regular video production companies by managing every aspect of video projects in a beautiful online experience . xxmaj with a growing global network of over 2,000 rated video professionals in over 50 countries managed by dedicated production success teams in 5 countries , 90 xxmaj seconds provides
1	xxmaj perry , xxmaj calvin xxmaj klein , xxmaj coach , xxmaj ellery , xxmaj xxunk , xxmaj xxunk , xxmaj gilt , xxmaj max xxmaj mara , xxmaj shopbop , xxmaj sneakerboy and xxmaj topshop . xxmaj the xxup pr xxmaj account xxmaj manager will be highly motivated and proactive . xxmaj the ideal candidate will have a proven track record with relevant case studies , sound knowledge
2	fill and cost per hire . xxmaj will be primarily responsible for maintaining xxup ats software to track applicants through the selection phase through to on - boarding , evaluating and choosing candidate sourcing options (i.e. linkedin , xxmaj indeed , xxmaj glassdoor , etc .) , and reviewing and adapting the recruiting processes from xxmaj headcount analysis through employee onboarding . xxmaj identify and source first
3	acetates , serving the xxmaj food , xxmaj pharmaceutical and xxmaj technical industries . xxmaj with two longstanding and fully automated manufacturing sites , located in xxmaj niagara xxmaj falls , xxup ny xxup usa , and xxmaj tiel , xxmaj the xxmaj netherlands , xxmaj niacet offers world - class quality products to a global market . xxmaj our products fill vital needs in a broad range of applications
4	the call centre industry in xxmaj mauritius . xxmaj operating since 1999 , xxup xxunk xxup bpo xxmaj services offers a range of contact centre / call centre solutions as well as xxup bpo services from its offices in xxmaj mauritius . xxup about xxup this jobwe are looking for potential candidates to promote online xxmaj chat and xxmaj customer xxmaj support
5	the country ; we also publish well - known professionals like xxmaj mario xxmaj batali , xxmaj gwyneth xxmaj paltrow , and xxmaj danny xxmaj meyer . xxmaj and we have partnerships with xxmaj whole xxmaj foods xxmaj market and xxmaj random xxmaj house . xxmaj food52 has been named the best food website by the xxmaj james xxmaj beard xxmaj foundation and xxup iacp , and has been featured

Creating text list of few columns data

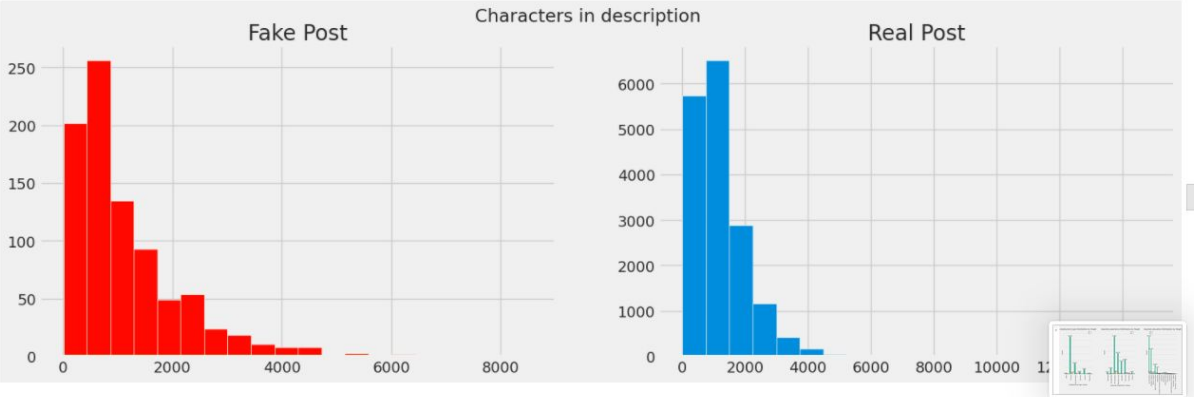


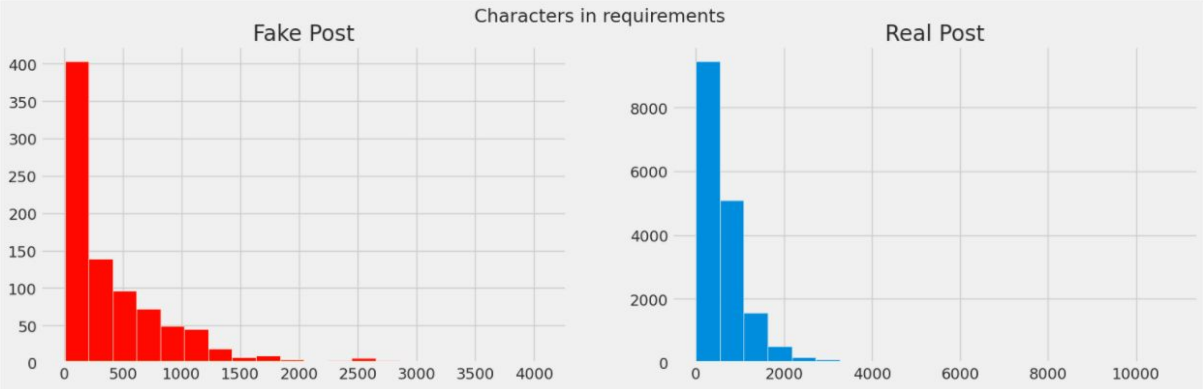
```
posx and posy should be finite values
```

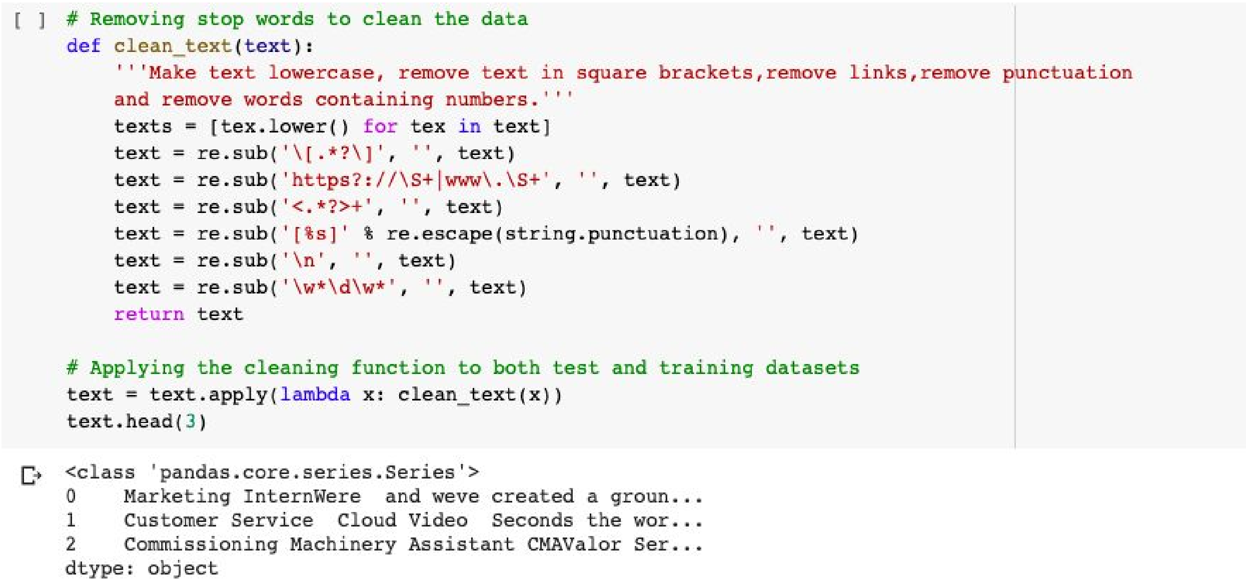


Visualizing categorical variable by target

Visualizing Posts with Characters with different column data







```
[ ] # Removing stop words to clean the data
    def clean_text(text):
        '''Make text lowercase, remove text in square brackets,remove links,remove punctuation
        and remove words containing numbers.'''
        texts = [tex.lower() for tex in text]
        text = re.sub('\[.*?\]', '', text)
        text = re.sub('https?://\S+|www\.\S+', '', text)
        text = re.sub('<.*?>+', '', text)
        text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
        text = re.sub('\n', '', text)
        text = re.sub('\w*\d\w*', '', text)
        return text

    # Applying the cleaning function to both test and training datasets
    text = text.apply(lambda x: clean_text(x))
    text.head(3)
```

```
<class 'pandas.core.series.Series'>
0    Marketing InternWere  and weve created a groun...
1    Customer Service  Cloud Video  Seconds the wor...
2    Commissioning Machinery Assistant CMAValor Ser...
dtype: object
```

Removing stop words to clean the data

3.4) XGBoost [7.1]**:**

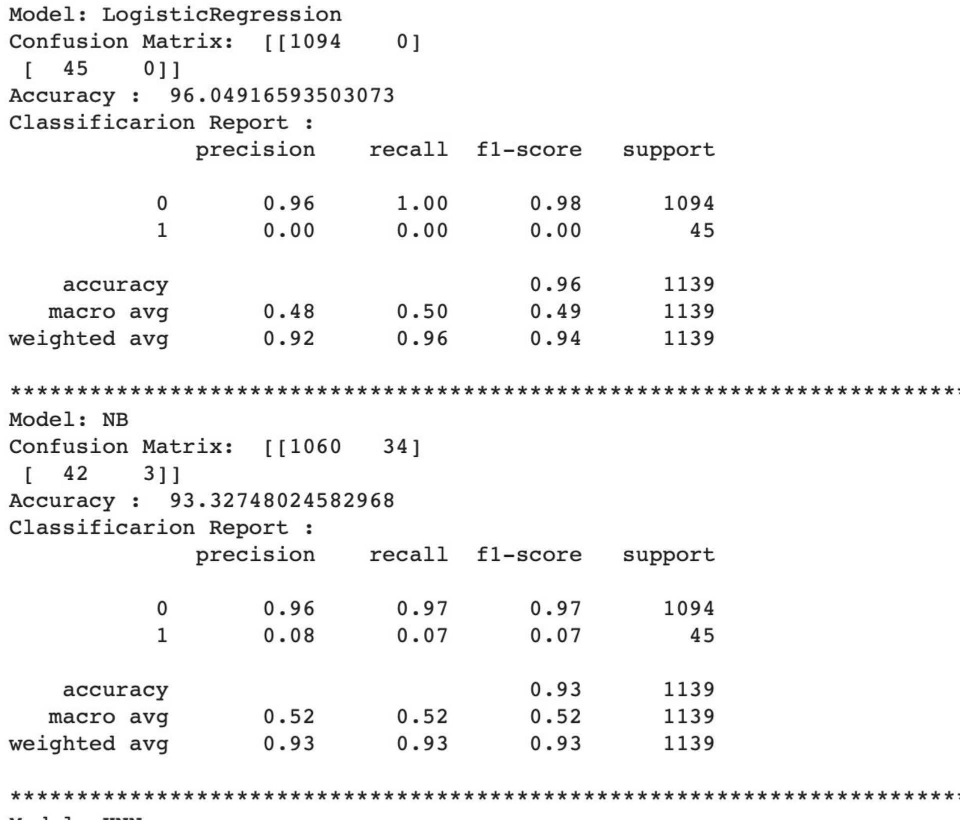
- XGBoost is a decision tree algorithm used for classification problems which uses a Gradient boosting framework.
- Decision trees are a good algorithm when the data is relatively small or medium and data is well structured.
- XGboost library is also available for interfacing with Command line interface, C++, Python, R, Java and many other languages.
- 3 Types of gradient boosting frameworks are available.
  - Gradient Boosting Machine
  - Stochastic Gradient Boosting
  - Regularized Gradient Boosting
  - Multiple additive regression trees
- Key features of XGboost library is that not only it provides a framework for Gradient Boosting but also optimizes the tree implementation through system optimization and algorithm improvement.
- Parallelization, distributed computing, cache optimization, tree pruning, sparse awareness, regularization, cross validations are some of the key features of XG Boost to improve system optimization and algorithm improvement
- For the implementation of XGBoost for Fake vs Real job posting classification, Two approaches can be followed.
  - Filtering the data with null values and converting descriptive columns with many unique values into numerical attributes. Finally generating a decision tree to efficiently classify the attributes based on best approximation of the result class using Gini impurity as measuring criteria
  - Second approach can be done using XG boost for text classification concentrating on the description of the job which will be neglected in the previous approach as having too many unique values. Text classification can be done by removing unnecessary words, punctuations, Stop words etc from the description. Vector space modelling can be used to convert these documents to vectors for classification. An XG boost pipeline is used in accordance with the vectorized document to classify the document to respective category. Different Vectorization techniques can be used in accordance with the XG Boost to obtain the result

4. Preliminary results

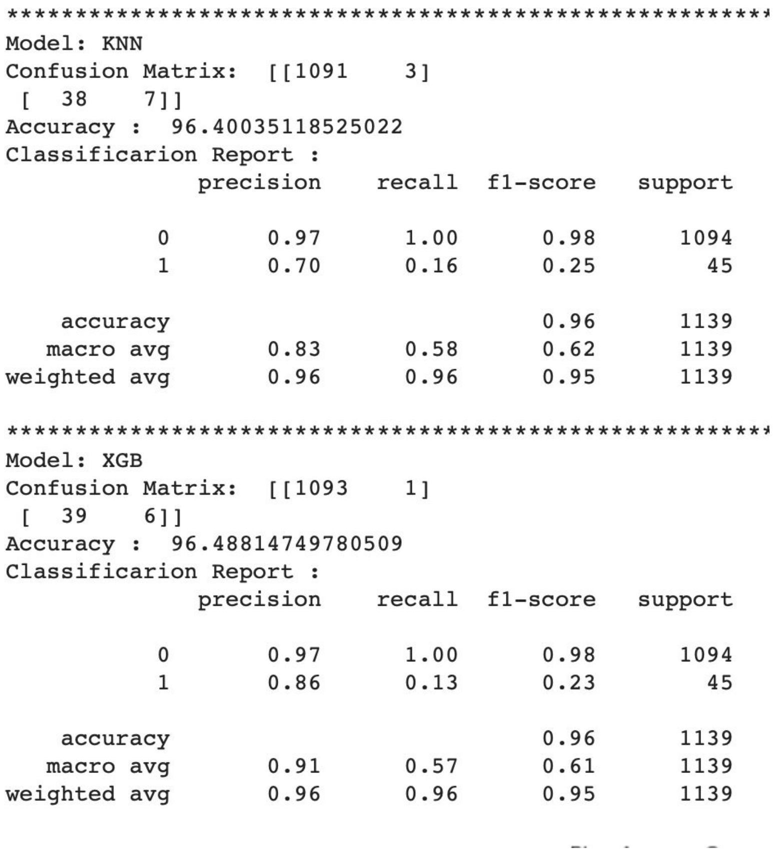
For preliminary analysis we wanted to see how different models (not necessarily great for text analysis) would perform on this kind of data. Thus, we first label encoded all of the columns, and tested models like Logistic Regression, K Nearest Neighbors and GaussianNB on the data.

An important thing to note for this is we did not do much data cleaning and preprocessing for this analysis. We did not even balance the dataset (which is extremely important for such datasets with high imbalance). The accuracy seems to be pretty high for these models, but we suspect that is because of the imbalance of the dataset. We aim to achieve a comparable or higher accuracy after balancing the data with the models we'll be working on.

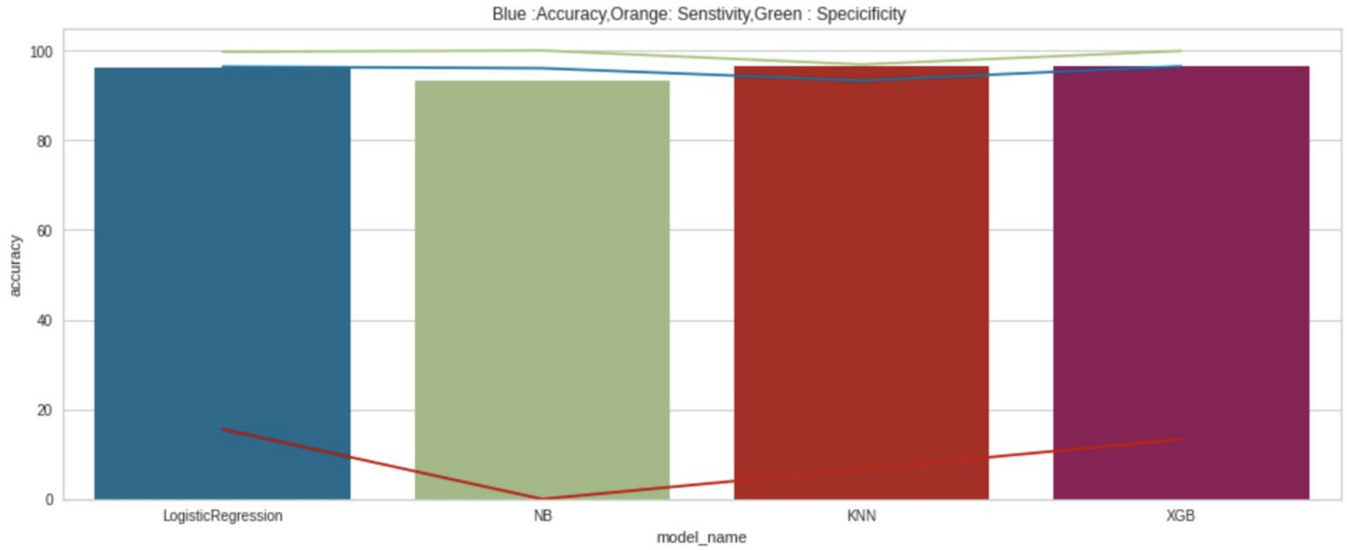
Following are the preliminary results we obtained for the above models. (These models are not part of the models we'll be working on).



```
Model: LogisticRegression
Confusion Matrix:  [[1094    0]
 [  45    0]]
Accuracy :  96.04916593503073
Classificarion Report :
              precision    recall  f1-score   support

           0       0.96      1.00      0.98      1094
           1       0.00      0.00      0.00        45

    accuracy                           0.96      1139
   macro avg       0.48      0.50      0.49      1139
weighted avg       0.92      0.96      0.94      1139

***********************************************************************
Model: NB
Confusion Matrix:  [[1060   34]
 [  42    3]]
Accuracy :  93.32748024582968
Classificarion Report :
              precision    recall  f1-score   support

           0       0.96      0.97      0.97      1094
           1       0.08      0.07      0.07        45

    accuracy                           0.93      1139
   macro avg       0.52      0.52      0.52      1139
weighted avg       0.93      0.93      0.93      1139

***********************************************************************
```



```
********************************************************
Model: KNN
Confusion Matrix:  [[1091    3]
 [  38    7]]
Accuracy :  96.40035118525022
Classificarion Report :
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      1094
           1       0.70      0.16      0.25        45

    accuracy                           0.96      1139
   macro avg       0.83      0.58      0.62      1139
weighted avg       0.96      0.96      0.95      1139

********************************************************
Model: XGB
Confusion Matrix:  [[1093    1]
 [  39    6]]
Accuracy :  96.48814749780509
Classificarion Report :
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      1094
           1       0.86      0.13      0.23        45

    accuracy                           0.96      1139
   macro avg       0.91      0.57      0.61      1139
weighted avg       0.96      0.96      0.95      1139
```



5. Repository

Following is the link to the Github repository for the CMPE 255 Data Mining project “Job posting analysis and fake vs real job posting detection”.

https://github.com/ShivanDesai/FakevsRealJobPostingDetection

The repository includes :

- Dataset
- Milestone - 2 report
- DataPreprocessing_EDA.ipynb
- Preliminary.ipynb
- CountVectorizer model(Fake_job_posting_Jasmine_NLP)

6. Participation

- **Sudarshan Aithal :** Algorithm Selection, Preprocessing the data, XGBoost classification.
- **Rutuja Hasurkar :** Exploratory Data Analysis(EDA), Random Forest Classification model implementation.
- **Shivan Desai:** Preprocessing, preliminary analysis, TF-IDF model.
- **Jasmine Akkal**: Data Preprocessing, Data Visualization, Data Cleaning, Countvectorizer Model, ULMFit Model

7. References

7.1. T Chen, C Guestrin
XGBoost: A Scalable Tree Boosting System
22nd ACM SIGKDD Int. Conf. 2016, 785.

7.2. Savan Patel
Machine Learning 101 - Random Forest Classifier

7.3 Leo Breiman and Adele Cutler
Random Forests, Salford Systems.