Industrial Classification

By: *Ahmed Abdulrazik Abduallah*

Amr Ahmed El Agoz

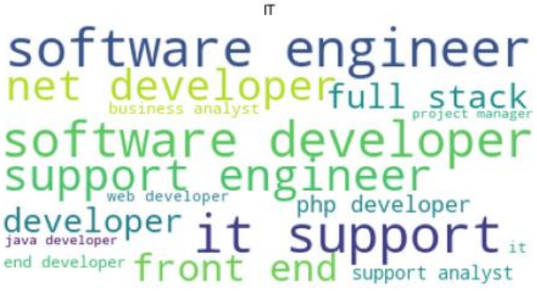
1- Problem Statement

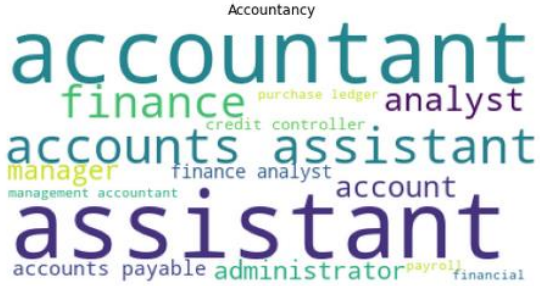
Given a dataset of 8587 rows about job titles and their industry, Our task is to classify the industry of the job given its title, the data contains unbalance classes and some misclassifications.

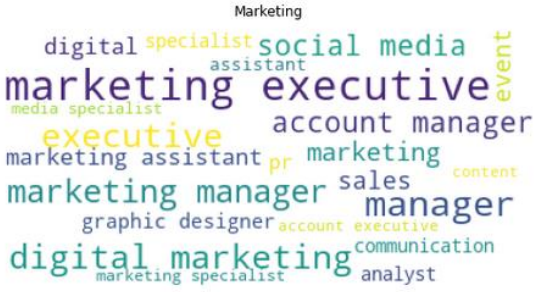
2- Data Cleaning:

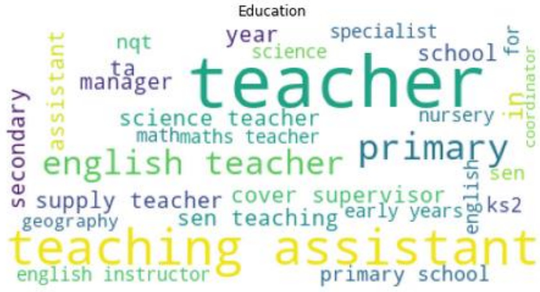
At data cleaning phase I made multiple assumption about the data

- Any character that is no from the English alphabet is useless that includes (numbers, non-English characters, punctuation)
- Any single character by its own is useless –some of those are result from the first assumption- like (‘k’, ‘c’).
- Decided not to remove all stop words because of “it” which is related to “IT” industrial, and some others may be useful, and because it is not required for this task as it is only job titles and not a usual talk.
- Some words are uninformative like (head, senior, junior, based, global, etc.) so I made a hand-crafted list of uninformative and some of stop words that may occur.









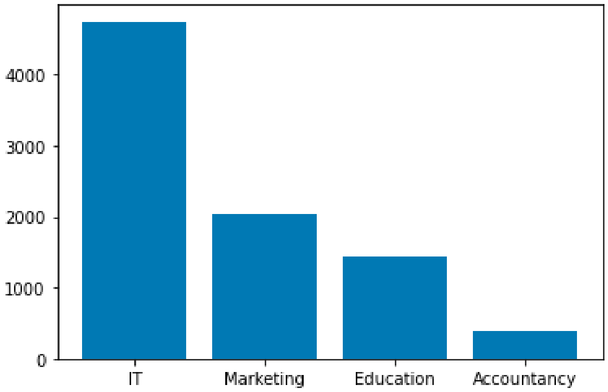
3- Data Splitting:

I split the data into 20% test 80% train with stratified splitting so that we keep the ratio of the classes in each set

4- Dealing with data imbalance:

Tried **SMOTE** from **imblearn** package to over sample the unbalanced classes, which is a data augmentation technic that generate data points near k neighbors. Followed by **RandomUnderSampler** the major classes as mentioned in reference, paper.

I tried other mentioned oversampling technics but **SMOTE** gave me the best results.



Then I tried training the classifiers with custom class weights

5- Tokenize the text data:

- Tried **TfidfVectorizer** and **CountVectorizer** decided to go with **CountVectorizer** under the assumption that TFIDF may remove important words like (engineer or developer) or give it low weight because they appear some many in the data.
- And decided to go with **n_gram** of (1,2) under the assumption that most of the jobs are 1 to 2 words

6- Choosing the Classifier

I tried several classifiers (**PassiveAggressiveClassifier**, **RandomForestClassifier**, **MultinomialNB**, **LogisticRegression**)

a- **PassiveAggressiveClassifier**: Online learning algorithm great with text data and big volumes of data
though our data is not big but vectorizer make it big in terms of features.

b- **RandomForestClassifier**: because trees in general can identify useful features.

c- **MultinomialNB**: -naïve bayes- fast, good accuracy, and great with the assumption that out features is independent from each other “web developer” = “developer web” both are IT industry.

d- **LogisticRegression**: In natural language processing, logistic regression is the baseline supervised machine learning algorithm for classification.

Trained these algorithms with cross validation, oversampling, undersampling, and these got results

Model	Average accuracy

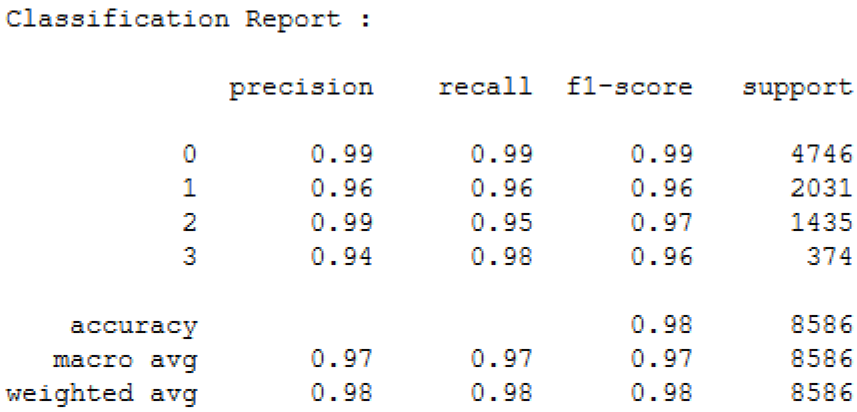
PassiveAggressiveClassifier	92.6%
RandomForestClassifier	92.1%
MultinomialNB	92.6%
LogisticRegression	92.8%

Then Combined them with **VotingClassifier** and got me an accuracy of 93.4% which is slightly better than all,

And tried again without oversampling but with custom class weights

Model	Average accuracy
PassiveAggressiveClassifier	92.8%
RandomForestClassifier	92.5%
MultinomialNB	93.5%
LogisticRegression	93.2%

Then Combined them with **VotingClassifier** and got me an accuracy of 93.7% which is slightly better than all, then train that **VotingClassifier** with the full dataset to use it in the RESTful api.

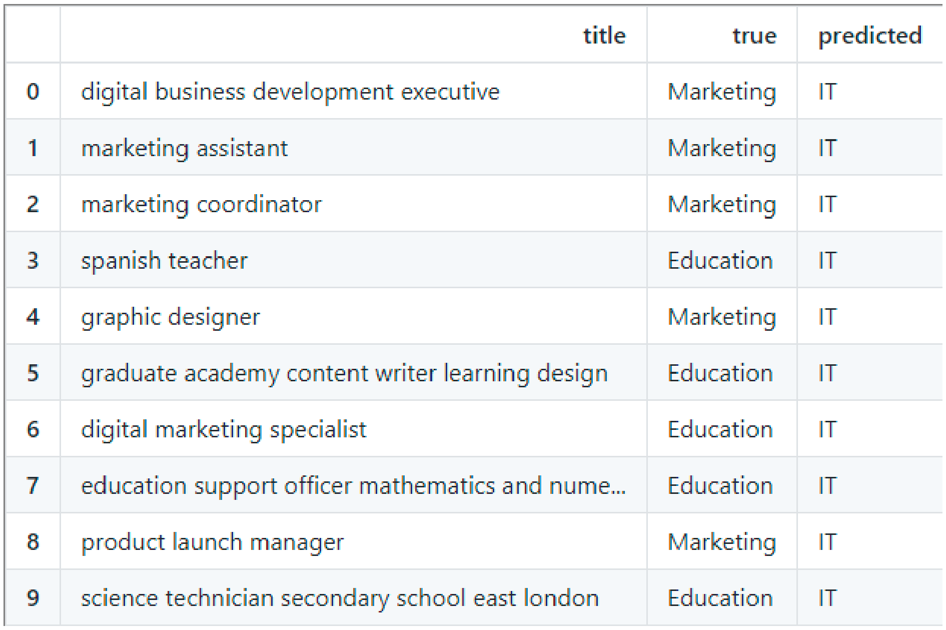


```
Classification Report :

              precision    recall  f1-score   support

           0       0.99      0.99      0.99      4746
           1       0.96      0.96      0.96      2031
           2       0.99      0.95      0.97      1435
           3       0.94      0.98      0.96       374

    accuracy                           0.98      8586
   macro avg       0.97      0.97      0.97      8586
weighted avg       0.98      0.98      0.98      8586
```

Classification report with **VotingClassifier** trained with full dataset

7- Error Analysis:

This part showed the model fails to predict the true labels



	title	true	predicted
0	digital business development executive	Marketing	IT
1	marketing assistant	Marketing	IT
2	marketing coordinator	Marketing	IT
3	spanish teacher	Education	IT
4	graphic designer	Marketing	IT
5	graduate academy content writer learning design	Education	IT
6	digital marketing specialist	Education	IT
7	education support officer mathematics and nume...	Education	IT
8	product launch manager	Marketing	IT
9	science technician secondary school east london	Education	IT

8- At the end saved the trained model using pickle package, to load it with flask api later.

9- ways to extend our model:

- referring to this paper per-class word vector normalization should help when dealing with imbalance dataset; in case of Naive Bayes
- a better text analysis and removing **stopwords** also may help
- NLP technics like **stemming**, **word embedding**, l**emmatization** may help simplify the data and the model

10- Limitations:

- though using **SMOTE** should help with imbalance data, we should consider gathering more data for "Accountancy" class
- better labeling should help too, as seen in Error analysis section