**Description:**

You can think of the job industry as the category or general field in which you work. On a job application, "industry" refers to a broad category under which a number of job titles can fall. For example, sales is an industry; job titles under this category can include sales associate, sales manager, manufacturing sales rep, pharmaceutical sales and so on.

**Requirements:**

I am asked to build a model using any Machine Learning classifier algorithm to classify job titles by the industry and provide insights on how the model works and answer the following questions:

- Which techniques you have used while cleaning the data if you have cleaned it?

- Why have you chosen this classifier? (E.g. I used Multinomial Naive Bayes because it is easy to interpret with text data and there are more than two outcomes).

- How do you deal with (Imbalance learning)?

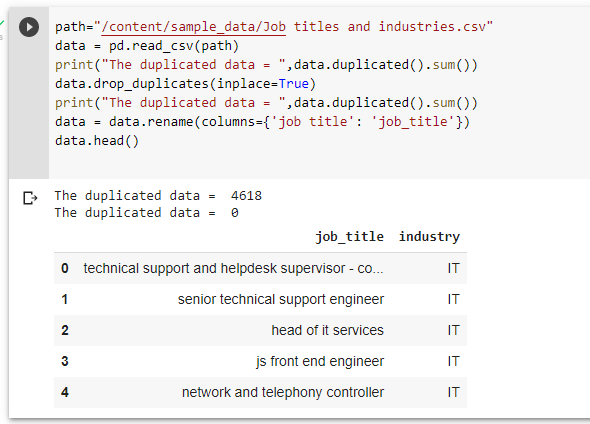
- How can you extend the model to have better performance?

- How do you evaluate your model? (i.e. accuracy, F1 score, Recall)

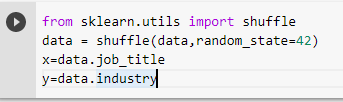
- What are the limitations of your methodology or Where does your approach fail? (e.g. your predictions are biased because you do not have enough data for a certain class)

To answer the previous questions I have made some insights to the dataset

* First to clean the dataset I check if it has duplicates I see 4618 duplicated row so I dropped it as it doesn’t hold any new piece of information but I know that as our dataset is Imbalance we can go to upsampling it using those duplicated row we dropped them but I treated it with different way as I will illustrate it later



* Then I realized that the classes is sorted so I shuffled the data to prevent my model to memorize any patterns in the data



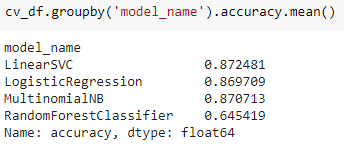
Well, as the classifiers and learning algorithms can not directly process the text documents in their original form, as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length. Therefore, during the preprocessing step, the texts are converted to a more manageable representation so for extracting features from text i used the bag of words model: a model where for each document, an industry in our case, the presence (and often the frequency) of words is taken into consideration, but the order in which they occur is ignored. I underline the previous sentence as it is a potential issue for our model. To do the previous approach I used

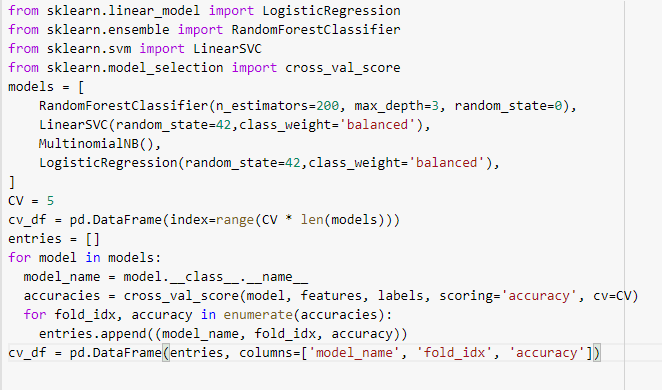
sklearn.feature\_extraction.text.TfidfVectorizer which calculate a measure called Term Frequency, Inverse Document Frequency.

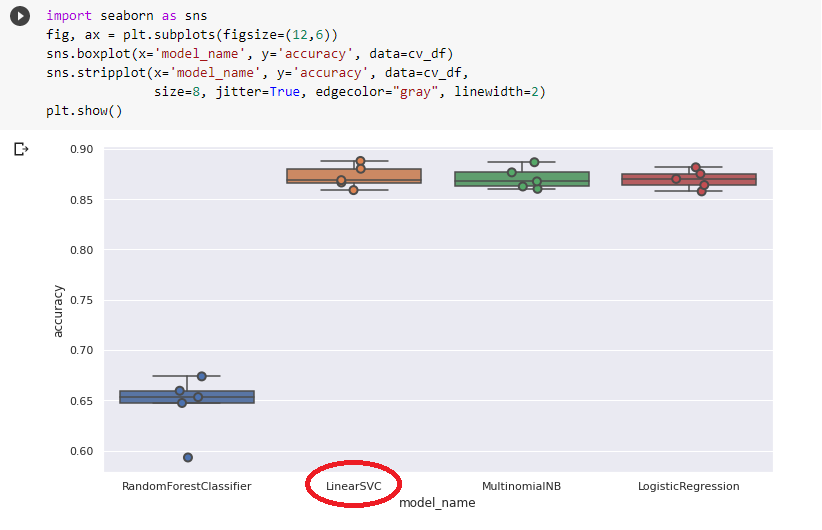
Then to select the model i experimented with different machine learning models, evaluate their accuracy and find the source of any potential issues I benchmarked the following four models:

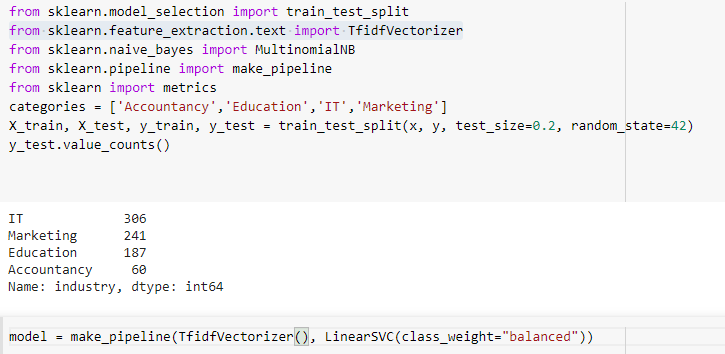
* Logistic Regression
* (Multinomial) Naive Bayes
* Linear Support Vector Machine
* Random Forest

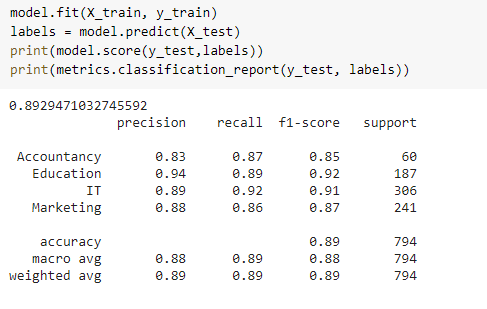
Then I plot the cross validation accuracy to choose the best one then I calculate the the mean accuracy of cross validation I found that linearsvc model was the best mean accuracy so I choose it







Then I split the data to train test split and made the pipeline of the model



* And I saw the report pretty good on our dataset as f1\_score is high for all the classes as they are close to 1
* To overcome the issue of Imbalance classes the model has Parameter

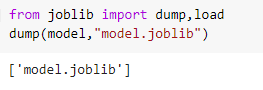
**class\_weight*dict or ‘balanced’, default=None***

* Set the parameter C of class i to class\_weight[i]\*C for SVC. If not given, all classes are supposed to have weight one. The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n\_samples / (n\_classes \* np.bincount(y)).

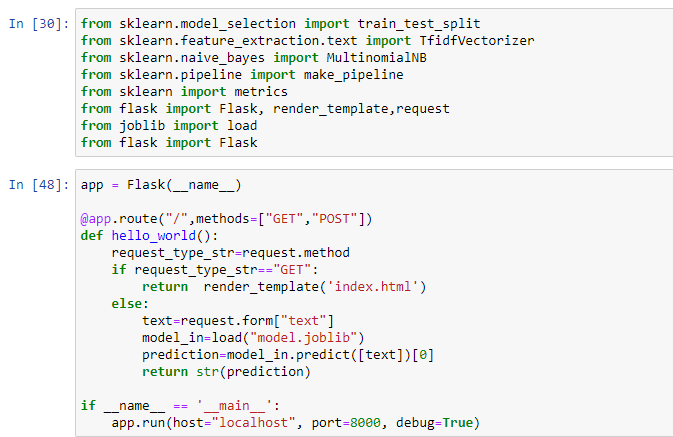
So I choose it “balanced” to give a larger weight to our imbalance class. Then I plot the confusion matrix to see what is going on



Then to use the model on restful APi I make the model as joblib file



So I used flask to make this restful APi on port 8000



And I created small html page

<form method="POST">

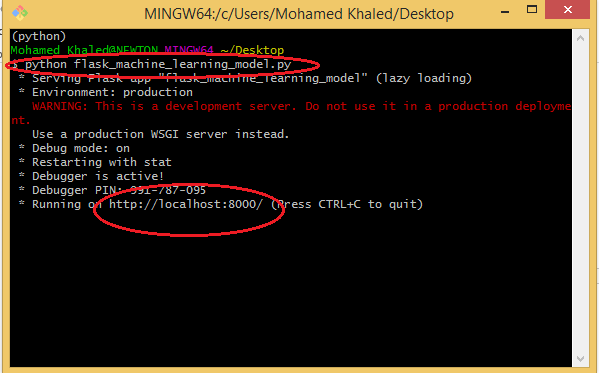
<label>input a job\_title</label>

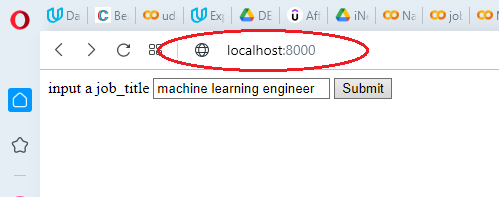
<input name="text">

<input type="submit">

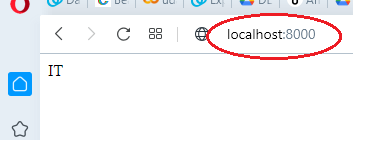
</form>

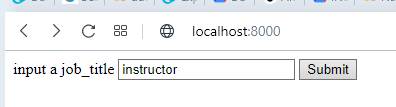
To get the title and response the industry and I fire the localhost using git bash

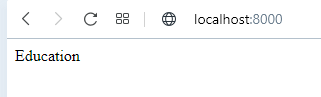


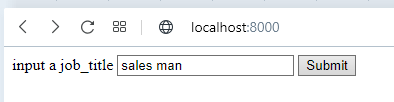


The response











According to this question

- How can you extend the model to have better performance?

This model as I mentioned earlier ignores the order of words as the TfidfVectorizer treats every word as an Independent feature so if you search for example for images of for chicago bulls team it will return images for Chicago separate from bulls and it doesn’t realize that chicago bulls team is a unique entity itself I once read that problem faced Google in early beginning before develop the pagerank algorithm

According to this question

-What are the limitations of your methodology or Where does your approach fail?

1. predictions are biased because you do not have enough data for a Accountancy class and the methodology fails when we input a job title have many words the model didn’t see it before
2. the weights for the low represented classes in our data
3. I evaluate the model using accuracy
4. We use TfidfVectorizer which Conventional algorithms are often biased towards the majority class, not taking the data distribution into consideration

**References:**

<https://stackoverflow.com/questions/15065833/imbalance-in-scikit-learn>

<https://flask.palletsprojects.com/en/2.0.x/quickstart/>