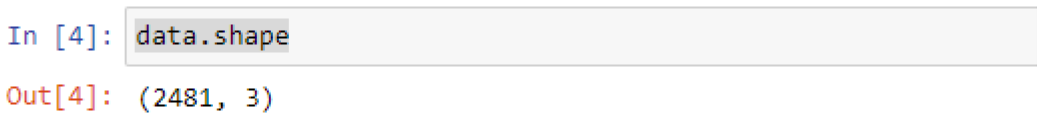
Assignment1 Report

The team share with me this JSON file for a group of categorized articles as I am supposed divide those articles into 3 groups: training data, validating data, testing data. To measure the accuracy of each algorithm, at this level you will measure the accuracy by the percent of matching only.

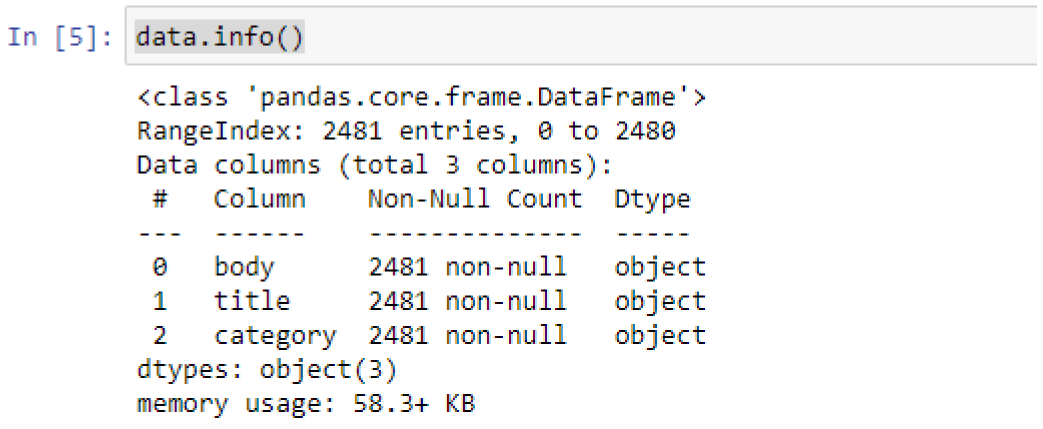
But in the case of these data, I decided to split it only in two groups: training data, and testing data. As it is no need to divide it in 3 groups, and I assumed that validating data is the same testing data.

So, I started with splitting data, the data size was



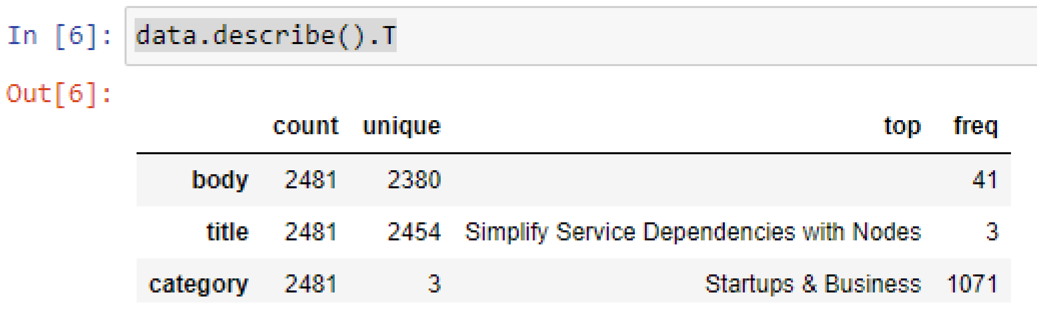
```
In [4]: data.shape

Out[4]: (2481, 3)
```



```
In [5]: data.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 2481 entries, 0 to 2480
        Data columns (total 3 columns):
         #   Column    Non-Null Count  Dtype
        ---  ------    --------------  -----
         0   body      2481 non-null   object
         1   title     2481 non-null   object
         2   category  2481 non-null   object
        dtypes: object(3)
        memory usage: 58.3+ KB
```



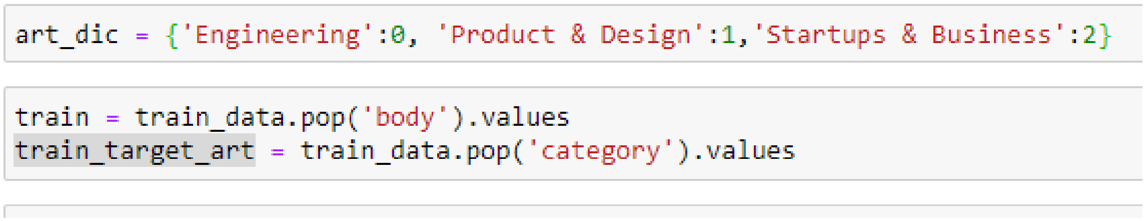
```
In [6]: data.describe().T

Out[6]:
```

	count	unique	top	freq
body	2481	2380		41
title	2481	2454	Simplify Service Dependencies with Nodes	3
category	2481	3	Startups & Business	1071

So, I made the training data from row 1 to row 1900, and the testing data is from row 1901 to the ending row 2481.

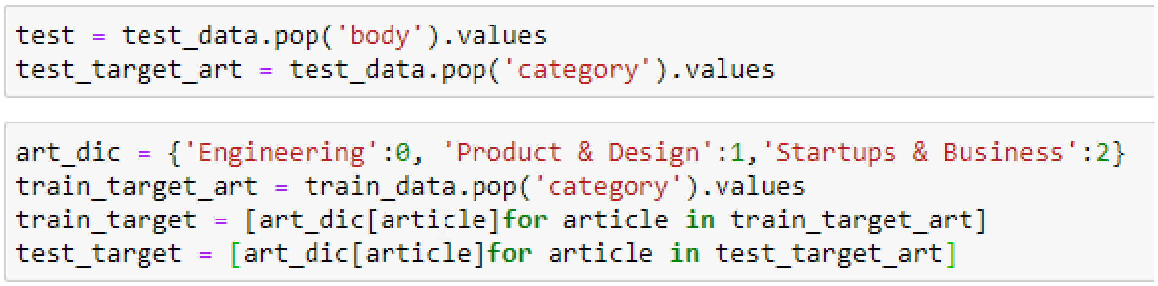
Then, I made the training data based on the 'body' feature, and the training target data based on the 'category' feature, after I had initiated a dictionary for category feature.



```
art_dic = {'Engineering':0, 'Product & Design':1,'Startups & Business':2}
```

```
train = train_data.pop('body').values
train_target_art = train_data.pop('category').values
```

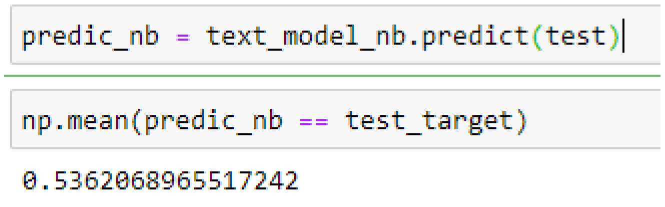
And I did the same for the testing data.



```
test = test_data.pop('body').values
test_target_art = test_data.pop('category').values
```

```
art_dic = {'Engineering':0, 'Product & Design':1,'Startups & Business':2}
train_target_art = train_data.pop('category').values
train_target = [art_dic[article]for article in train_target_art]
test_target = [art_dic[article]for article in test_target_art]
```

Then I started to build the NB model, by CountVectorizer, tfidfTransformer, and MultinomialNb, after that I put all in the pipeline which produced the predicted NB, and by comparison with the test_target, the mean was 53.62%

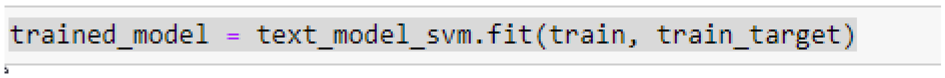


```
predic_nb = text_model_nb.predict(test)
```

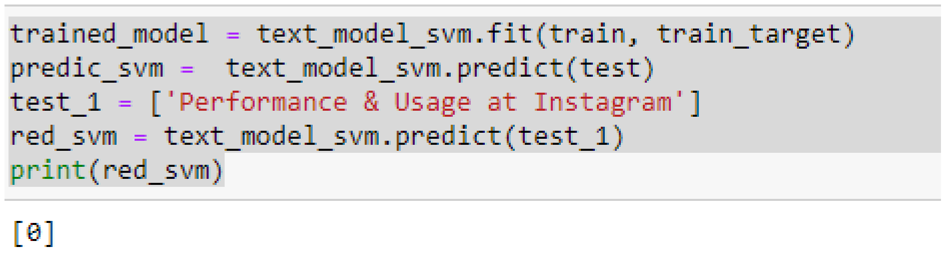
```
np.mean(predic_nb == test_target)
```

```
0.5362068965517242
```

I repeated building the model, but this time by LinearSVC, and put the pipeline again, and this time the accuracy by comparison predicted data with target data was 87.58%.

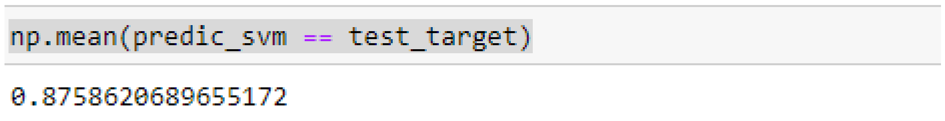


```
trained_model = text_model_svm.fit(train, train_target)
```



```
trained_model = text_model_svm.fit(train, train_target)
predic_svm =  text_model_svm.predict(test)
test_1 = ['Performance & Usage at Instagram']
red_svm = text_model_svm.predict(test_1)
print(red_svm)
```

```
[0]
```



```
np.mean(predic_svm == test_target)
```

```
0.8758620689655172
```

Which tells us that the SVM model gives higher accuracy for this kind of articles classifications.