

Lab 06

In Github it is possible to languages0.zip, languages1.zip, languages2.zip, languages3.zip, languages4.zip, could be accessed in the following folder of github: <https://github.com/masterfloss/lang/raw/main/> .

1. Comment the following code

```
#import libraries
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split

file0='https://github.com/masterfloss/lang/raw/main/languages0.zip'

df=pd.read_csv(file0)
x = np.array(df["text"])
y = np.array(df["language"])

cv = CountVectorizer()
X = cv.fit_transform(x)

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)

model = MultinomialNB()

model.fit(X_train, y_train)
model.score(X_test, y_test)
```

1. Analyze the following code:

```
val=input("expression:")
x=cv.transform([val]).toarray()
model.predict(x)[0]
```

2.1. Create a function called detect that allows guessing the language given a text. Must return a value to be stored in a variable.

2.2. Create another function that asks the user for an expression and calls the previous function.

1. How many sentences are there for each language?
2. Import read all the files to dataframes and merge all. Use the method concatenate.
3. How many sentences start with character "A".

4. Save the dataframe in several parts.

some code that may be useful :

```
dataframe.groupby(by='dataframecolumn').count()
```

```
dataframe[dataframe['dataframecolumn'].astype(str).str[0]=="A"]
```

```
pd.concat([df1,df2])
```

```
np.array_split(df,n)
```

If you want to know more:

The CountVectorizer provides a way to tokenize a collection of text documents and build a vocabulary of known words. An encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document. The encoded vectors can then be used directly with a machine learning algorithm.

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

<https://www.analyticsvidhya.com/blog/2021/06/part-5-step-by-step-guide-to-master-nlp-text-vectorization-approaches/>

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html