In [1]:
```
import pandas as pd
from nltk.tokenize import word_tokenize
from collections import defaultdict
from nltk.corpus import wordnet as wn
from nltk.stem import WordNetLemmatizer
from nltk import pos_tag
from nltk.corpus import stopwords
from sklearn.pipeline import make_pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
import joblib
```

In [2]:
```
corpus = pd.read_csv(r"./Sample Data 1.csv")
```

In [3]:
```
# Remove blank rows
corpus['Description'].dropna(inplace=True)
# Change all text to lower case
corpus['Description'] = [entry.lower() for entry in corpus['Description']]
# Tokenization
corpus['Description']= [word_tokenize(entry) for entry in corpus['Description']]
corpus.head()
```

Out[3]:

	Genre	Description
0	Places & Travel	[the, budget-minded, traveler, podcast, is, yo...
1	Food	[the, official, podcast, of, bourbon, ., featu...
2	Social Sciences	[..., our, most, human, experiences, ...]
3	News & Politics	[a, show, about, politics, with, no, agenda, ,...
4	Careers	[made, for, profit, is, a, podcast, where, we,...

In [4]:
```
# Remove stop words, non-numeric and perfom Word Stemming/Lemmenting.
tag_map = defaultdict(lambda : wn.NOUN)
tag_map['J'] = wn.ADJ
tag_map['V'] = wn.VERB
tag_map['R'] = wn.ADV
for index,entry in enumerate(corpus['Description']):
    Final_words = []
    word_Lemmatized = WordNetLemmatizer()
    for word, tag in pos_tag(entry):
        if word not in stopwords.words('english') and word.isalpha():
            word_Final = word_Lemmatized.lemmatize(word,tag_map[tag[0]])
            Final_words.append(word_Final)
    corpus.loc[index,'description_final'] = str(Final_words)
```

In [5]:
```
# Build the model
NBmodel = make_pipeline(TfidfVectorizer(), naive_bayes.MultinomialNB())
SVMmodel = make_pipeline(TfidfVectorizer(), svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto', probability=True))
# Train the model using the training data
NBmodel.fit(corpus['description_final'], corpus['Genre'])
SVMmodel.fit(corpus['description_final'], corpus['Genre'])
```

Out[5]:
```
Pipeline(steps=[('tfidfvectorizer', TfidfVectorizer()),
                ('svc', SVC(gamma='auto', kernel='linear', probability=True))])
```

In [6]:
```
# Save Models
joblib.dump(NBmodel, "NBmodel")
joblib.dump(SVMmodel, "SVMmodel")
```

Out[6]:
```
['SVMmodel']
```

In [ ]: