PRN: 21070126039 Name: Jainil Patel Batch: AI/ML A2

Git Repository : GitRepo

In [ ]:
```
import nltk
from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_tokenize
from nltk.stem import WordNetLemmatizer
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
from nltk.corpus import stopwords
nltk.download("all")
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
!unzip /usr/share/nltk_data/corpora/wordnet.zip -d /usr/share/nltk_data/corpora/
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

```
[nltk_data] Downloading package stopwords to /usr/share/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /usr/share/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /usr/share/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading collection 'all'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /usr/share/nltk_data...
[nltk_data]    |   Package abc is already up-to-date!
[nltk_data]    | Downloading package alpino to /usr/share/nltk_data...
[nltk_data]    |   Package alpino is already up-to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package averaged_perceptron_tagger is already up-
[nltk_data]    |       to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger_ru to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Unzipping
[nltk_data]    |       taggers/averaged_perceptron_tagger_ru.zip.
[nltk_data]    | Downloading package basque_grammars to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package basque_grammars is already up-to-date!
[nltk_data]    | Downloading package bcp47 to /usr/share/nltk_data...
[nltk_data]    | Downloading package biocreative_ppi to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package biocreative_ppi is already up-to-date!
[nltk_data]    | Downloading package bllip_wsj_no_aux to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package bllip_wsj_no_aux is already up-to-date!
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package book_grammars is already up-to-date!
[nltk_data]    | Downloading package brown to /usr/share/nltk_data...
[nltk_data]    |   Package brown is already up-to-date!
[nltk_data]    | Downloading package brown_tei to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package brown_tei is already up-to-date!
[nltk_data]    | Downloading package cess_cat to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package cess_cat is already up-to-date!
[nltk_data]    | Downloading package cess_esp to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package cess_esp is already up-to-date!
[nltk_data]    | Downloading package chat80 to /usr/share/nltk_data...
[nltk_data]    |   Package chat80 is already up-to-date!
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package city_database is already up-to-date!
[nltk_data]    | Downloading package cmudict to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package comparative_sentences to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Unzipping corpora/comparative_sentences.zip.
[nltk_data]    | Downloading package comtrans to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package comtrans is already up-to-date!
[nltk_data]    | Downloading package conll2000 to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package conll2000 is already up-to-date!
[nltk_data]    | Downloading package conll2002 to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package conll2002 is already up-to-date!
[nltk_data]    | Downloading package conll2007 to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package conll2007 is already up-to-date!
[nltk_data]    | Downloading package crubadan to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package crubadan is already up-to-date!
[nltk_data]    | Downloading package dependency_treebank to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package dependency_treebank is already up-to-date!
[nltk_data]    | Downloading package dolch to /usr/share/nltk_data...
[nltk_data]    |   Unzipping corpora/dolch.zip.
[nltk_data]    | Downloading package europarl_raw to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package europarl_raw is already up-to-date!
[nltk_data]    | Downloading package extended_omw to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    | Downloading package floresta to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package floresta is already up-to-date!
[nltk_data]    | Downloading package framenet_v15 to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Unzipping corpora/framenet_v15.zip.
[nltk_data]    | Downloading package framenet_v17 to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Unzipping corpora/framenet_v17.zip.
[nltk_data]    | Downloading package gazetteers to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package gazetteers is already up-to-date!
[nltk_data]    | Downloading package genesis to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package genesis is already up-to-date!
[nltk_data]    | Downloading package gutenberg to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package gutenberg is already up-to-date!
[nltk_data]    | Downloading package ieer to /usr/share/nltk_data...
[nltk_data]    |   Package ieer is already up-to-date!
[nltk_data]    | Downloading package inaugural to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package inaugural is already up-to-date!
[nltk_data]    | Downloading package indian to /usr/share/nltk_data...
[nltk_data]    |   Package indian is already up-to-date!
[nltk_data]    | Downloading package jeita to /usr/share/nltk_data...
[nltk_data]    |   Package jeita is already up-to-date!
[nltk_data]    | Downloading package kimmo to /usr/share/nltk_data...
[nltk_data]    |   Package kimmo is already up-to-date!
[nltk_data]    | Downloading package knbc to /usr/share/nltk_data...
[nltk_data]    |   Package knbc is already up-to-date!
[nltk_data]    | Downloading package large_grammars to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package large_grammars is already up-to-date!
[nltk_data]    | Downloading package lin_thesaurus to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package lin_thesaurus is already up-to-date!
[nltk_data]    | Downloading package mac_morpho to
[nltk_data]    |     /usr/share/nltk_data...
```

```
[nltk_data]    |   Package mac_morpho is already up-to-date!
[nltk_data]    | Downloading package machado to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package machado is already up-to-date!
[nltk_data]    | Downloading package masc_tagged to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package masc_tagged is already up-to-date!
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package maxent_ne_chunker is already up-to-date!
[nltk_data]    | Downloading package maxent_treebank_pos_tagger to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package maxent_treebank_pos_tagger is already up-
[nltk_data]    |       to-date!
[nltk_data]    | Downloading package moses_sample to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package moses_sample is already up-to-date!
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package movie_reviews is already up-to-date!
[nltk_data]    | Downloading package mte_teip5 to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package mte_teip5 is already up-to-date!
[nltk_data]    | Downloading package mwa_ppdb to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Unzipping misc/mwa_ppdb.zip.
[nltk_data]    | Downloading package names to /usr/share/nltk_data...
[nltk_data]    |   Package names is already up-to-date!
[nltk_data]    | Downloading package nombank.1.0 to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    | Downloading package nonbreaking_prefixes to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Unzipping corpora/nonbreaking_prefixes.zip.
[nltk_data]    | Downloading package nps_chat to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package nps_chat is already up-to-date!
[nltk_data]    | Downloading package omw to /usr/share/nltk_data...
[nltk_data]    |   Package omw is already up-to-date!
[nltk_data]    | Downloading package omw-1.4 to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    | Downloading package opinion_lexicon to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package opinion_lexicon is already up-to-date!
[nltk_data]    | Downloading package panlex_swadesh to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    | Downloading package paradigms to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package paradigms is already up-to-date!
[nltk_data]    | Downloading package pe08 to /usr/share/nltk_data...
[nltk_data]    |   Unzipping corpora/pe08.zip.
[nltk_data]    | Downloading package perluniprops to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Unzipping misc/perluniprops.zip.
[nltk_data]    | Downloading package pil to /usr/share/nltk_data...
[nltk_data]    |   Package pil is already up-to-date!
[nltk_data]    | Downloading package pl196x to /usr/share/nltk_data...
[nltk_data]    |   Package pl196x is already up-to-date!
[nltk_data]    | Downloading package porter_test to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package porter_test is already up-to-date!
[nltk_data]    | Downloading package ppattach to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package ppattach is already up-to-date!
[nltk_data]    | Downloading package problem_reports to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package problem_reports is already up-to-date!
[nltk_data]    | Downloading package product_reviews_1 to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package product_reviews_1 is already up-to-date!
[nltk_data]    | Downloading package product_reviews_2 to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package product_reviews_2 is already up-to-date!
[nltk_data]    | Downloading package propbank to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package propbank is already up-to-date!
[nltk_data]    | Downloading package pros_cons to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package pros_cons is already up-to-date!
[nltk_data]    | Downloading package ptb to /usr/share/nltk_data...
[nltk_data]    |   Package ptb is already up-to-date!
[nltk_data]    | Downloading package punkt to /usr/share/nltk_data...
[nltk_data]    |   Package punkt is already up-to-date!
[nltk_data]    | Downloading package qc to /usr/share/nltk_data...
[nltk_data]    |   Package qc is already up-to-date!
[nltk_data]    | Downloading package reuters to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package reuters is already up-to-date!
[nltk_data]    | Downloading package rslp to /usr/share/nltk_data...
[nltk_data]    |   Package rslp is already up-to-date!
[nltk_data]    | Downloading package rte to /usr/share/nltk_data...
[nltk_data]    |   Package rte is already up-to-date!
[nltk_data]    | Downloading package sample_grammars to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package sample_grammars is already up-to-date!
[nltk_data]    | Downloading package semcor to /usr/share/nltk_data...
[nltk_data]    |   Package semcor is already up-to-date!
[nltk_data]    | Downloading package senseval to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package senseval is already up-to-date!
[nltk_data]    | Downloading package sentence_polarity to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package sentence_polarity is already up-to-date!
[nltk_data]    | Downloading package sentiwordnet to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package sentiwordnet is already up-to-date!
[nltk_data]    | Downloading package shakespeare to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package shakespeare is already up-to-date!
[nltk_data]    | Downloading package sinica_treebank to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package sinica_treebank is already up-to-date!
[nltk_data]    | Downloading package smultron to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package smultron is already up-to-date!
[nltk_data]    | Downloading package snowball_data to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package snowball_data is already up-to-date!
[nltk_data]    | Downloading package spanish_grammars to
[nltk_data]    |     /usr/share/nltk_data...
```

```
[nltk_data]    |   Package spanish_grammars is already up-to-date!
[nltk_data]    | Downloading package state_union to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package state_union is already up-to-date!
[nltk_data]    | Downloading package stopwords to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package stopwords is already up-to-date!
[nltk_data]    | Downloading package subjectivity to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package subjectivity is already up-to-date!
[nltk_data]    | Downloading package swadesh to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package swadesh is already up-to-date!
[nltk_data]    | Downloading package switchboard to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package switchboard is already up-to-date!
[nltk_data]    | Downloading package tagsets to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package tagsets is already up-to-date!
[nltk_data]    | Downloading package timit to /usr/share/nltk_data...
[nltk_data]    |   Package timit is already up-to-date!
[nltk_data]    | Downloading package toolbox to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package toolbox is already up-to-date!
[nltk_data]    | Downloading package treebank to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package treebank is already up-to-date!
[nltk_data]    | Downloading package twitter_samples to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package twitter_samples is already up-to-date!
[nltk_data]    | Downloading package udhr to /usr/share/nltk_data...
[nltk_data]    |   Package udhr is already up-to-date!
[nltk_data]    | Downloading package udhr2 to /usr/share/nltk_data...
[nltk_data]    |   Package udhr2 is already up-to-date!
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package unicode_samples is already up-to-date!
[nltk_data]    | Downloading package universal_tagset to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package universal_tagset is already up-to-date!
[nltk_data]    | Downloading package universal_treebanks_v20 to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package universal_treebanks_v20 is already up-to-
[nltk_data]    |       date!
[nltk_data]    | Downloading package vader_lexicon to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package vader_lexicon is already up-to-date!
[nltk_data]    | Downloading package verbnet to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package verbnet is already up-to-date!
[nltk_data]    | Downloading package verbnet3 to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Unzipping corpora/verbnet3.zip.
[nltk_data]    | Downloading package webtext to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package webtext is already up-to-date!
[nltk_data]    | Downloading package wmt15_eval to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Unzipping models/wmt15_eval.zip.
[nltk_data]    | Downloading package word2vec_sample to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package word2vec_sample is already up-to-date!
[nltk_data]    | Downloading package wordnet to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet2021 to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    | Downloading package wordnet2022 to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Unzipping corpora/wordnet2022.zip.
[nltk_data]    | Downloading package wordnet31 to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    | Downloading package wordnet_ic to
[nltk_data]    |     /usr/share/nltk_data...
[nltk_data]    |   Package wordnet_ic is already up-to-date!
[nltk_data]    | Downloading package words to /usr/share/nltk_data...
[nltk_data]    |   Package words is already up-to-date!
[nltk_data]    | Downloading package ycoe to /usr/share/nltk_data...
[nltk_data]    |   Package ycoe is already up-to-date!
[nltk_data]    |
[nltk_data]  Done downloading collection all
[nltk_data] Downloading package stopwords to /usr/share/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /usr/share/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Archive:  /usr/share/nltk_data/corpora/wordnet.zip
   creating: /usr/share/nltk_data/corpora/wordnet/
  inflating: /usr/share/nltk_data/corpora/wordnet/lexnames
  inflating: /usr/share/nltk_data/corpora/wordnet/data.verb
  inflating: /usr/share/nltk_data/corpora/wordnet/index.adv
  inflating: /usr/share/nltk_data/corpora/wordnet/adv.exc
  inflating: /usr/share/nltk_data/corpora/wordnet/index.verb
  inflating: /usr/share/nltk_data/corpora/wordnet/cntlist.rev
  inflating: /usr/share/nltk_data/corpora/wordnet/data.adj
  inflating: /usr/share/nltk_data/corpora/wordnet/index.adj
  inflating: /usr/share/nltk_data/corpora/wordnet/LICENSE
  inflating: /usr/share/nltk_data/corpora/wordnet/citation.bib
  inflating: /usr/share/nltk_data/corpora/wordnet/noun.exc
  inflating: /usr/share/nltk_data/corpora/wordnet/verb.exc
  inflating: /usr/share/nltk_data/corpora/wordnet/README
  inflating: /usr/share/nltk_data/corpora/wordnet/index.sense
  inflating: /usr/share/nltk_data/corpora/wordnet/data.noun
  inflating: /usr/share/nltk_data/corpora/wordnet/data.adv
  inflating: /usr/share/nltk_data/corpora/wordnet/index.noun
  inflating: /usr/share/nltk_data/corpora/wordnet/adj.exc
```

1. Introduction

Content

1.2. Importing the libraries

In [ ]:
```
# Importing the libraries
import pandas as pd
```

1.3. Importing the dataset

In [ ]:
```
# Importing the dataset
# Define the column names for the data
dataset = pd.read_csv('/kaggle/input/multilingual-lyrics-for-genre-classification/train.csv')
df = pd.DataFrame(dataset)
df.head()
```

Out[ ]:

	Artist	Song	Genre	Language	Lyrics
0	12 stones	world so cold	Rock	en	It starts with pain, followed by hate\nFueled ...
1	12 stones	broken	Rock	en	Freedom!\nAlone again again alone\nPatiently w...
2	12 stones	3 leaf loser	Rock	en	Biting the hand that feeds you, lying to the v...
3	12 stones	anthem for the underdog	Rock	en	You say you know just who I am\nBut you can't ...
4	12 stones	adrenaline	Rock	en	My heart is beating faster can't control these...

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['Artist', 'Song', 'Genre', 'Language', 'Lyrics'], dtype='object')
```

In [ ]:
```
df.shape
```

Out[ ]:
```
(290183, 5)
```

In [ ]:
```
df.describe()
```

Out[ ]:

	Artist	Song	Genre	Language	Lyrics
count	290183	290182	290183	290179	290148
unique	11152	164357	10	33	249297
top	elvis presley	intro	Rock	en	Instrumental
freq	1611	163	121404	250197	540

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 290183 entries, 0 to 290182
Data columns (total 5 columns):
 #   Column    Non-Null Count   Dtype 
---  ------    --------------   ----- 
 0   Artist    290183 non-null  object
 1   Song      290182 non-null  object
 2   Genre     290183 non-null  object
 3   Language  290179 non-null  object
 4   Lyrics    290148 non-null  object
dtypes: object(5)
memory usage: 11.1+ MB
```

In [ ]:
```
df['Artist'].value_counts()
```

Out[ ]:
```
elvis presley           1611
chris brown             1239
elvis costello           923
ella fitzgerald          874
the rolling stones       820
                        ...
the gregory brothers       1
flamingokvintetten         1
fjeld                      1
danceplaycreate            1
crawdad republic           1
Name: Artist, Length: 11152, dtype: int64
```

In [ ]:
```
len(df['Artist'].unique())
```

Out[ ]:
```
11152
```

In [ ]:
```
df['Genre'].value_counts()
```

Out[ ]:
```
Rock          121404
Pop           108714
Metal          20291
Jazz           13545
Folk            8644
Indie           8449
R&B             2793
Hip-Hop         2240
Electronic      2213
Country         1890
Name: Genre, dtype: int64
```

In [ ]:
```
len(df['Genre'].unique())
```

Out[ ]:
```
10
```

In [ ]:
```
df['Language'].value_counts()
```

Out[ ]:
```
en    250197
pt     30102
es      3892
ro      1184
it       808
id       737
fr       644
de       478
sw       304
tl       241
so       229
cy       226
ca       137
tr       116
nl       116
sk        98
hr        97
no        93
sl        77
af        77
da        71
sv        61
et        58
fi        54
pl        24
cs        17
sq        15
hu        10
vi         7
ru         4
lt         2
lv         2
ko         1
Name: Language, dtype: int64
```

In [ ]:
```
len(df['Language'].unique())
```

Out[ ]:
```
34
```

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
Artist       0
Song         1
Genre        0
Language     4
Lyrics      35
dtype: int64
```

In [ ]:
```
df.dropna(inplace=True)
```

In [ ]:
```
df['Lyrics'].isna().sum()
```

Out[ ]:
```
0
```

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
Artist      0
Song        0
Genre       0
Language    0
Lyrics      0
dtype: int64
```

In [ ]:
```
df.shape
```

Out[ ]:
```
(290143, 5)
```

In [ ]:
```
int(df.shape[0]/20)
```

Out[ ]:
```
14507
```

In [ ]:
```
num_samples_per_class = 5000

sliced_df = pd.DataFrame(columns=df.columns)

unique_classes = df['Genre'].unique()
for class_label in unique_classes:
    class_df = df[df['Genre'] == class_label]
    if len(class_df) >= num_samples_per_class:
```

```
        class_samples = class_df.sample(n=num_samples_per_class, random_state=42)
    else:
        class_samples = class_df  # Use all available samples if there are fewer than num_samples_per_class
    sliced_df = pd.concat([sliced_df, class_samples])


df = sliced_df.copy()
```

In [ ]:
```
df.shape
```

Out[ ]:
```
(39136, 5)
```

In [ ]:
```
df['Genre'].value_counts()
```

Out[ ]:
```
Rock          5000
Metal         5000
Pop           5000
Indie         5000
Folk          5000
Jazz          5000
R&B           2793
Hip-Hop       2240
Electronic    2213
Country       1890
Name: Genre, dtype: int64
```

2. Data Preprocessing

In [ ]:
```
df = df[df['Genre'].isin(['Rock', 'Jazz', 'Hip-Hop', 'Metal', 'Country'])]
```

In [ ]:
```
df['Genre'].value_counts()
```

Out[ ]:
```
Rock       5000
Metal      5000
Jazz       5000
Hip-Hop    2240
Country    1890
Name: Genre, dtype: int64
```

In [ ]:
```
# Preprocessing the data using NLTK

# Importing the libraries
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
```

In [ ]:
```
df.shape
```

Out[ ]:
```
(19130, 5)
```

2.1. Lemmatization and Tokenization

In [ ]:
```
# Initialize the lemmatizer
lemmatizer = WordNetLemmatizer()

# Defining a function to tokenize and lemmetize the text

def tokenize_and_lemmatize(text):
    tokens = word_tokenize(text)
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]
    return " ".join(lemmatized_tokens)
```

In [ ]:
```
# Applying the function to the text column
df['Lyrics'] = df['Lyrics'].apply(tokenize_and_lemmatize)
```

3. Data Cleaning

3.1 Remove stopwords, Remove symbols, Remove URLs

In [ ]:
```
# Data Cleansing: Remove stopwords, remove symbols, remove URLs

# Importing the libraries
import re
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))
```

In [ ]:
```
# Defining a function to clean the text
def clean_text(text):
    # Remove URLs
    text = re.sub(r'http\S+', '', text)
    # Remove symbols and numbers
    text = re.sub(r'[^\w\s]', '', text)
    # Remove stopwords
    text = " ".join([word for word in text.split() if word.lower() not in stop_words])
    return text
```

In [ ]:
```
# Applying the clean text function to the text column
df['Lyrics'] = df['Lyrics'].apply(clean_text)
# Displaying the first 5 rows of the dataset
df.head()
```

Out[ ]:

	Artist	Song	Genre	Language	Lyrics
256544	mott the hoople	pearl 'n' roy (england)	Rock	en	shut clean chimney kid 1974 shake fist make ol...
145115	the rolling stones	back of my hand	Rock	en	hear preacher corner Ranting like crazy man sa...
8907	bryan adams	is your mama gonna miss ya?	Rock	en	might well bin blind Thought owned world time ...
141431	peter bjorn and john	may seem macabre	Rock	en	darkenwide misty eye say five break night eye ...
33998	mana	róbame el alma	Rock	es	robame el alma bebela embriagate hasta caer am...

4 Converting text to vectors

4.1 Using ConuntVectorizer

In [ ]:
```
# Using countvectorizer to convert text to vectors
from sklearn.feature_extraction.text import CountVectorizer

# Initialize CountVectorizer
count_vectorizer = CountVectorizer()

# Fit and transform the text column
count_vectors = count_vectorizer.fit_transform(df['Lyrics'])


# Print the shape and a small sample of the output
print("Shape of CountVectorizer output:", count_vectors.shape)
print("Sample of CountVectorizer output:\n", count_vectors[0:5])
```

```
Shape of CountVectorizer output: (19130, 69061)
Sample of CountVectorizer output:
   (0, 54980)   1
  (0, 12066)    1
  (0, 11429)    1
  (0, 33600)    1
  (0, 264)      1
  (0, 54297)    1
  (0, 23492)    1
  (0, 37066)    2
  (0, 42962)    1
  (0, 63205)    1
  (0, 66424)    1
  (0, 2223)     2
  (0, 42259)    3
  (0, 46711)    1
  (0, 67667)    2
  (0, 39013)    1
  (0, 22255)    1
  (0, 35687)    1
  (0, 66350)    1
  (0, 41493)    1
  (0, 25890)    2
  (0, 7573)     1
  (0, 7549)     1
  (0, 25159)    1
  (0, 42323)    3
  :     :
  (4, 20352)    1
  (4, 61167)    1
  (4, 48634)    1
  (4, 46660)    1
  (4, 6866)     1
  (4, 20035)    3
  (4, 21314)    4
  (4, 31837)    3
  (4, 24955)    3
  (4, 55078)    1
  (4, 20813)    2
  (4, 44871)    1
  (4, 13916)    1
  (4, 34390)    1
  (4, 42266)    1
  (4, 15768)    1
  (4, 44561)    1
  (4, 16907)    1
  (4, 15014)    1
  (4, 35912)    2
  (4, 42876)    1
  (4, 37458)    1
  (4, 16908)    2
  (4, 15015)    2
  (4, 61537)    2
```

4.2 Using TF-IDF

In [ ]:
```
# Using TF-IDF to convert text to vectors
from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()

# Fit and transform the text column
tfidf_vectors = tfidf_vectorizer.fit_transform(df['Lyrics'])

# Print the shape and a small sample of the output
print("Shape of TFIDFVectorizer output:", tfidf_vectors.shape)
print("Sample of TFIDFVectorizer output:\n", tfidf_vectors[0:5])
```

```
Shape of TFIDFVectorizer output: (19130, 69061)
Sample of TFIDFVectorizer output:
   (0, 58009)   0.047784765123869415
  (0, 39072)    0.06177086187245119
  (0, 54328)    0.06398968124415798
  (0, 13703)    0.03613040631108714
  (0, 9463)     0.019003846356325203
  (0, 34625)    0.08219135471602551
  (0, 33971)    0.05197233025201602
  (0, 27206)    0.05905048693610614
  (0, 55421)    0.08837602843935535
  (0, 54889)    0.053562731170563274
  (0, 60228)    0.04389923385057285
  (0, 22423)    0.047558596787565174
  (0, 11358)    0.05584967419216082
  (0, 66934)    0.03393440902681408
  (0, 6874)     0.029496042460212396
  (0, 7555)     0.04831803468641895
  (0, 67254)    0.033231784462485016
  (0, 1426)     0.0747831525819462
  (0, 61311)    0.016881504600495197
  (0, 26453)    0.0365194499910823
  (0, 60872)    0.02135534645414828
  (0, 60370)    0.03929409639958651
  (0, 28803)    0.05299711662554344
  (0, 10344)    0.04173449488650284
  (0, 36251)    0.027496076851097322
  :     :
  (4, 20035)    0.11828044916790567
  (4, 6866)     0.030905247686823015
  (4, 46660)    0.03587313575799543
  (4, 48634)    0.016900905395643172
  (4, 61167)    0.026171734027163374
  (4, 20352)    0.02507193460509276
  (4, 2503)     0.027351567055516546
  (4, 28236)    0.025528736445325776
  (4, 45779)    0.03456818436139286
  (4, 9484)     0.03942681638930189
  (4, 36631)    0.1287520790548046
  (4, 62912)    0.2643157223533351
  (4, 13070)    0.2379129537983728
  (4, 58681)    0.035166032038062456
  (4, 38711)    0.14227586942292322
  (4, 39561)    0.07885363277860379
  (4, 20639)    0.03942681638930189
  (4, 2812)     0.03942681638930189
  (4, 9574)     0.03587313575799543
  (4, 28125)    0.06563047103330662
  (4, 20221)    0.03942681638930189
  (4, 6157)     0.03942681638930189
  (4, 2672)     0.49714459900928626
  (4, 19944)    0.44316821681745516
  (4, 51448)    0.5125486130609246
```

4.3 Using Word2Vec

In [ ]:
```python
# Using word2vec to convert text to vectors
from gensim.models import Word2Vec

# Tokenize the text column
tokenized_text = df['Lyrics'].apply(lambda x: x.split())

# Train a Word2Vec model
word2vec_model = Word2Vec(tokenized_text, window=5, min_count=1, workers=4)
word2vec_model.train(tokenized_text, total_examples=len(tokenized_text), epochs=10)

# Create Word2Vec vectors for the text column
word2vec_vectors = tokenized_text.apply(lambda x: [word2vec_model.wv[word] for word in x])

# Print the shape and a small sample of the output
print("Sample of Word2Vec output:", word2vec_vectors.head())
```

```
Sample of Word2Vec output: 256544    [[-1.1274967, -0.23310012, -1.1081858, 0.88278...
145115    [[-0.7397139, 0.4824181, -3.3763597, -2.000244...
8907      [[1.0459611, -0.36676323, 0.91696805, 1.696775...
141431    [[-0.013266287, -0.030382048, -0.0126411095, -...
33998     [[-0.2751627, 0.17716298, -0.003995462, -0.214...
Name: Lyrics, dtype: object
```

4.4 Using GoogleNews Word2Vec

In [ ]:
```python
# Using GoogleNews word2vec to convert text to vectors
from gensim.models import KeyedVectors

# Load the GoogleNews Word2Vec model
google_w2v_path = '/kaggle/input/googlenewsvectorsnegative300/GoogleNews-vectors-negative300.bin'
google_w2v_model = KeyedVectors.load_word2vec_format(google_w2v_path, binary=True)

# Create GoogleNews Word2Vec vectors for the text column
google_w2v_vectors = tokenized_text.apply(lambda x: [google_w2v_model[word] for word in x if word in google_w2v_model])

# Print the shape and a small sample of the output
print("Sample of GoogleNews Word2Vec output:", google_w2v_vectors.head())
```

```
Sample of GoogleNews Word2Vec output: 256544    [[-0.14160156, 0.024536133, 0.49023438, -0.016...
145115    [[0.06689453, -0.19726562, 0.12988281, 0.05029...
8907      [[0.22753906, 0.12011719, 0.068359375, 0.32421...
141431    [[0.29101562, 0.17675781, 0.037109375, -0.1083...
33998     [[-0.012023926, 0.20410156, 0.22949219, -0.009...
Name: Lyrics, dtype: object
```

Since the vectorization techniques produce different formats of data, we'll need to handle them separately. We'll start with Logistic Regression using CountVectorizer and TFIDFVectorizer.

4.5 Lable Encoding

In [ ]:
```python
y = df['Genre']
```

In [ ]:
```python
y.value_counts()
```

Out[ ]:
```
Rock       5000
Metal      5000
Jazz       5000
Hip-Hop    2240
Country    1890
Name: Genre, dtype: int64
```

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
# Encode genre labels
label_encoder = LabelEncoder()
genre_encoded = label_encoder.fit_transform(y)
```

In [ ]:
```
unique_labels = df['Genre'].unique()
unique_encoded = label_encoder.transform(unique_labels)

# Create a DataFrame to display the unique labels and their encoded values
df_encoded = pd.DataFrame({'Unique Labels': unique_labels, 'Encoded Labels': unique_encoded})

# Print the DataFrame
df_encoded
```

Out[ ]:

	Unique Labels	Encoded Labels
0	Rock	4
1	Metal	3
2	Jazz	2
3	Hip-Hop	1
4	Country	0

5. Applying Machine Learning Models

5.1 Logistics Regression

5.1.1 Logistics Regression with CountVectorizer

In [ ]:
```
# Logistic Regression with CountVectorizer

# Importing the libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(count_vectors,genre_encoded, test_size=0.2, random_state=42)

# Initialize and train the Logistic Regression model
lr_model_count = LogisticRegression(max_iter=1000)
lr_model_count.fit(X_train, y_train)

# Make predictions and print the classification report
y_pred_count = lr_model_count.predict(X_test)
print("Classification Report for Logistic Regression with CountVectorizer:\n", classification_report(y_test, y_pred_count))
```

```
Classification Report for Logistic Regression with CountVectorizer:
              precision    recall  f1-score   support

           0       0.38      0.37      0.37       362
           1       0.78      0.70      0.74       431
           2       0.67      0.74      0.70       999
           3       0.69      0.67      0.68      1039
           4       0.47      0.45      0.46       995

    accuracy                           0.61      3826
   macro avg       0.60      0.59      0.59      3826
weighted avg       0.61      0.61      0.61      3826
```

5.1.2 Logistics Regression with TFIDFVectorizer

In [ ]:
```
# Logistic Regression with TFIDFVectorizer
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(tfidf_vectors, genre_encoded, test_size=0.2, random_state=42)

# Initialize and train the Logistic Regression model
lr_model_tfidf = LogisticRegression(max_iter=1000)
lr_model_tfidf.fit(X_train, y_train)

# Make predictions and print the classification report
y_pred_tfidf = lr_model_tfidf.predict(X_test)
print("Classification Report for Logistic Regression with TFIDFVectorizer:\n", classification_report(y_test, y_pred_tfidf))
```

```
Classification Report for Logistic Regression with TFIDFVectorizer:
              precision    recall  f1-score   support

           0       0.54      0.24      0.33       362
           1       0.85      0.70      0.77       431
           2       0.67      0.73      0.70       999
           3       0.70      0.77      0.73      1039
           4       0.50      0.55      0.52       995

    accuracy                           0.64      3826
   macro avg       0.65      0.60      0.61      3826
weighted avg       0.64      0.64      0.64      3826
```

For using word2vec and Google news word2vec we need to convert the vectors into different format since these vectors are a list of vectors for each token in a document.

we need to aggregate them into a single vector for each document before feeding them into the models. A common approach is to take the mean of all vectors for each document.

5.1.3 Logistics Regression with Word2Vec

In [ ]:
```
# Logistic Regression with Word2Vec
# Importing the libraries
import numpy as np

# Function to calculate the mean vector for each document
def mean_vector(words):
    # Filter out words that are not in the Word2Vec model's vocabulary
    valid_words = [word for word in words if word in word2vec_model.wv]
    if valid_words:
        # Calculate the mean vector for valid words
        vectors = [word2vec_model.wv[word] for word in valid_words]
        return np.mean(vectors, axis=0)
    else:
        # Return a zero vector if no valid words are found
        return np.zeros(word2vec_model.vector_size)
```

In [ ]:
```
# Apply the function to the tokenized text
word2vec_mean_vectors = np.array(tokenized_text.apply(mean_vector).tolist())

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(word2vec_mean_vectors, genre_encoded, test_size=0.2, random_state=42)

# Initialize and train the Logistic Regression model
lr_model_word2vec = LogisticRegression(max_iter=1000)
lr_model_word2vec.fit(X_train, y_train)

# Make predictions and print the classification report
y_pred_word2vec = lr_model_word2vec.predict(X_test)
print("Classification Report for Logistic Regression with Word2Vec:\n", classification_report(y_test, y_pred_word2vec))
```

```
Classification Report for Logistic Regression with Word2Vec:
              precision    recall  f1-score   support

           0       0.28      0.11      0.16       362
           1       0.70      0.69      0.70       431
           2       0.63      0.69      0.65       999
           3       0.67      0.72      0.70      1039
           4       0.45      0.48      0.46       995

    accuracy                           0.59      3826
   macro avg       0.55      0.54      0.53      3826
weighted avg       0.57      0.59      0.57      3826
```

5.1.4 Logistics Regression with GoogleNews Word2Vec

In [ ]:
```
# Function to calculate the mean vector for each document
def mean_vector_google_w2v(words):
    # Filter out words that are not in the GoogleNews Word2Vec model's vocabulary
    valid_words = [word for word in words if word in google_w2v_model]
    if valid_words:
        # Calculate the mean vector for valid words
        vectors = [google_w2v_model[word] for word in valid_words]
        return np.mean(vectors, axis=0)
    else:
        # Return a zero vector if no valid words are found
        return np.zeros(google_w2v_model.vector_size)
```

In [ ]:
```
# Apply the function to the tokenized text
google_w2v_mean_vectors = np.array(tokenized_text.apply(mean_vector_google_w2v).tolist())

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(google_w2v_mean_vectors, genre_encoded, test_size=0.2, random_state=42)

# Initialize and train the Logistic Regression model
lr_model_google_w2v = LogisticRegression(max_iter=1000)
lr_model_google_w2v.fit(X_train, y_train)

# Make predictions and print the classification report
y_pred_google_w2v = lr_model_google_w2v.predict(X_test)
print("Classification Report for Logistic Regression with GoogleNews Word2Vec:\n", classification_report(y_test, y_pred_google_w2v))
```

```
Classification Report for Logistic Regression with GoogleNews Word2Vec:
              precision    recall  f1-score   support

           0       0.50      0.19      0.28       362
           1       0.76      0.69      0.72       431
           2       0.64      0.71      0.67       999
           3       0.69      0.75      0.72      1039
           4       0.48      0.50      0.49       995

    accuracy                           0.62      3826
   macro avg       0.61      0.57      0.58      3826
weighted avg       0.61      0.62      0.61      3826
```

5.2 SVM

Using SGV on CountVectorizer as count vectorizer makes such a large dataset which is difficult to process by local host or even google collab so Going with SGD classifier

5.2.1 SGD Classifier & GridSearchCV with CountVectorizer

In [ ]:
```
# Importing the SGD Classifier
from sklearn.linear_model import SGDClassifier

# Split the data into training and testing sets and training the model
X_train, X_test, y_train, y_test = train_test_split(count_vectors, genre_encoded, test_size=0.2, random_state=42)
sgd_model_count = SGDClassifier(loss='hinge', max_iter=10000)
sgd_model_count.fit(X_train, y_train)

# Make predictions and printing the classification report
y_pred_count_sgd = sgd_model_count.predict(X_test)
print("Classification Report for SGD Classifier with CountVectorizer:\n", classification_report(y_test, y_pred_count_sgd))
```

```
Classification Report for SGD Classifier with CountVectorizer:
              precision    recall  f1-score   support

           0       0.39      0.37      0.38       362
           1       0.71      0.78      0.74       431
           2       0.68      0.71      0.69       999
           3       0.69      0.64      0.66      1039
           4       0.45      0.46      0.46       995

    accuracy                           0.60      3826
   macro avg       0.58      0.59      0.59      3826
weighted avg       0.60      0.60      0.60      3826
```

In [ ]:
```python
# Check the distribution of the target values
target_distribution = df['Lyrics'].value_counts()
print(target_distribution)
```

```
Instrumental
43
heart sad lonely sigh dear nt seen body soul spend day longing wondering wronging tell mean body soul ca nt believe hard conceive turn away romance pretending look like ending Un
less could one chance prove dear life wreck making know taking gladly surrender body soul life wreck making know taking gladly surrender body soul
39
hate see evening sun go hate see evening sun go Cause baby gone left town Feelin tomorrow like feel today feelin tomorrow like feel today pack truck make giveaway St Louis woman
diamond ring Pulls man around wa nt man love would gone nowhere nowhere got St Louis blue blue man got heart like rock cast sea else would nt gone far love baby like school boy l
ove pie Like Kentucky colonel love mint n rye love man till day die
19
nt know love Til learned meaning blue loved love lose nt know love nt know lip hurt kissed pay cost flipped heart lost nt know love know lost heart fear thought reminiscing lip t
aste tear Lose taste kissing nt know heart burn love live yet never dy faced dawn sleepless eye nt know love
18
Speak low speak love summer day withers away soon soon Speak low speak love moment swift like ship adrift swept apart soon Speak low darling speak low Love spark lost dark soon s
oon feel wherever go tomorrow near tomorrow always soon Time old love brief Love pure gold time thief late darling late curtain descends evrything end soon soon wait darling wait
speak low Speak love soon
18

..
Years ago came back ha hardest attack burning fire brain could feel deadly flame Tales magic head dark dream fall mind hold key fortune feel strange thing mind stone Ambar must f
ind take chance day Chorus Halloween wizard crown take Halloween Halloween return Halloween servant Blind cross realm death gate open wide close dungeon deep place hide time chan
ged ha fight magic spell fall Illusions falling mind key find ritual ha begun Chorus battle ha begun warlock time ha come pay price pay nt see sign loosing life die forever life
Halloween wizard crown take Halloween Halloween return Halloween Halloween wizard crown take
1
fragile game play ghost yesterday ca nt let go never say goodbye trace remains stone mark graf memory thought could deny much lose pain put carelessness left dark blood may wash
away scar never fade least know somehow made mark dark light Nothing left nothing right x3 Nothing right x2
1
Little act little deed little mind breed breed kind Attitude gratitude let shit carry every day muthafuckas bury think dress act strange unchange word exchange Take deep breath t
ry leave shit unsaid step stage unleash hatred Chorus Know fuck talkin Know fuck talkin Know fuck talkin Know fuck talkin way way Hypocrites politics democracy Governments overra
n hypocrisy go go business sellin soul ask forgiveness think everybody better come clean raise wrong happening Take deep breath try leave shit unsaid step stage unleash hatred Ch
orus Get back Get back muthafucker get back
1
spit cut tie hold tight time ha come realize thing loved despise life time life time picture wall around nt feel never felt way around nt feel spit justify time ha come realize t
hing loved despise life time life time picture wall around nt feel never felt way around nt feel end know faking try get back know faking end life time life time rise
1
rodeo rodeo riding several year ol Cheyenne Houston never cause much fear let tell one chill soul happens December snowin cold Oklahoma City building concrete toughest stock men
gather compete point tallied stock ha brought got two hour cause start eight oclock Finals NFR Series sport Hey rookie take ten head got heart think tough cowboy find end final w
histle blow stock pen coliseum quiet except sound cowboy getting ready workman ground cowboy ask got draw ol Necklace pluck old DoubleOught tension mounting crowd start pouring s
hiver go like cold cold wind hear horse comin runnin alleyway snortin blowin men shut sliding gate Finals NFR much time ago enough time Anthem grand entry going sit bronc ready w
ait hear chute gate open look gate horse come boiling blow roof come another kicking like curly wolf hear chute bos holler yelling crowd say one ahead better get screwed Finals N
FR nod head daze horse go boiling run spurrs neck drag em mind blur eye seeing red flank catch slam back rump bang head somewhere background hear buzzer sound hand jerked riggin
crash ground stumble foot stagger wall wonder really worth Finals NFR     1
Name: Lyrics, Length: 17367, dtype: int64
```

5.2.2 SVM Classifier with TFIDFVectorizer

In [ ]:
```python
# Importing the libraries
from sklearn.svm import LinearSVC
# Split the TFIDFVectorizer data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(tfidf_vectors, genre_encoded, test_size=0.2, random_state=42)

# Initialize LinearSVC model
linear_svc_model_tfidf = LinearSVC(max_iter=10000)

# Train the model
linear_svc_model_tfidf.fit(X_train, y_train)

# Make predictions on the test set
y_pred_tfidf_svc = linear_svc_model_tfidf.predict(X_test)

# Print the classification report
print("Classification Report for LinearSVC with TFIDFVectorizer:\n", classification_report(y_test, y_pred_tfidf_svc))
```

```
Classification Report for LinearSVC with TFIDFVectorizer:
              precision    recall  f1-score   support

           0       0.48      0.35      0.41       362
           1       0.84      0.75      0.80       431
           2       0.68      0.74      0.71       999
           3       0.71      0.75      0.73      1039
           4       0.51      0.50      0.50       995

    accuracy                           0.65      3826
   macro avg       0.64      0.62      0.63      3826
weighted avg       0.64      0.65      0.64      3826
```

5.2.3 SVM Classifier & GridSearchCV with Word2Vec

In [ ]:
```python
# Trying SGD Model for word2vec

# Importing the SGD Classifier
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import RandomizedSearchCV # Import RandomizedSearchCV
# Split the Word2Vec mean vectors into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(word2vec_mean_vectors, genre_encoded, test_size=0.2, random_state=42)

from sklearn.metrics import classification_report

# Hyperparameters to search (reduced grid)
param_grid = {
    'alpha': [1e-3, 1e-2], # Reduced number of options
    'max_iter': [1000, 5000], # Reduced number of options
    'penalty': ['l2'], # Keeping only 'l2'
    'loss': ['hinge']  # Keeping only 'hinge'
}

# Create a RandomizedSearchCV object with the SGD Classifier (you can switch to GridSearchCV if desired)
```

```
# Here, n_iter is set to 4, so only 4 random combinations will be tried
random_search_word2vec_sgd = RandomizedSearchCV(SGDClassifier(), param_grid, n_iter=4, cv=5, verbose=1, n_jobs=-1)

# Fit the model to the training data
random_search_word2vec_sgd.fit(X_train, y_train)

# Print the best parameters and score
print("Best parameters found:", random_search_word2vec_sgd.best_params_)

# Make predictions on the test set using the best model
y_pred_word2vec_sgd_random = random_search_word2vec_sgd.predict(X_test)

# Print the classification report for the model with the best hyperparameters
print("Classification Report for SGD Classifier with Word2Vec (RandomizedSearchCV):\n", classification_report(y_test, y_pred_word2vec_sgd_random))
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
Best parameters found: {'penalty': 'l2', 'max_iter': 1000, 'loss': 'hinge', 'alpha': 0.001}
Classification Report for SGD Classifier with Word2Vec (RandomizedSearchCV):
              precision    recall  f1-score   support

           0       0.24      0.11      0.15       362
           1       0.51      0.78      0.62       431
           2       0.52      0.77      0.62       999
           3       0.63      0.76      0.69      1039
           4       0.57      0.16      0.25       995

    accuracy                           0.55      3826
   macro avg       0.49      0.52      0.47      3826
weighted avg       0.54      0.55      0.50      3826
```

5.2.4 SGD Classifier with GoogleNews Word2Vec

In [ ]:
```
# Split the GoogleNews Word2Vec mean vectors into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(google_w2v_mean_vectors, genre_encoded, test_size=0.2, random_state=42)

# Initialize SGD Classifier
sgd_model_google_w2v = SGDClassifier(loss='hinge', max_iter=10000)

# Train the model
sgd_model_google_w2v.fit(X_train, y_train)

# Make predictions on the test set
y_pred_google_w2v_sgd = sgd_model_google_w2v.predict(X_test)

# Print the classification report
print("Classification Report for SGD Classifier with GoogleNews Word2Vec:\n", classification_report(y_test, y_pred_google_w2v_sgd))
```

```
Classification Report for SGD Classifier with GoogleNews Word2Vec:
              precision    recall  f1-score   support

           0       0.57      0.07      0.12       362
           1       0.68      0.75      0.71       431
           2       0.59      0.75      0.66       999
           3       0.67      0.78      0.72      1039
           4       0.49      0.41      0.44       995

    accuracy                           0.60      3826
   macro avg       0.60      0.55      0.53      3826
weighted avg       0.59      0.60      0.58      3826
```

5.3 Random Forest

5.3.1 Random Forest with CountVectorizer

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier

# Split the CountVectorizer data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(count_vectors,genre_encoded, test_size=0.2, random_state=42)

# Initialize Random Forest Classifier with modified hyperparameters
rf_model_count = RandomForestClassifier(n_estimators=50, max_depth=10, max_features='sqrt', random_state=42)

# Train the model
rf_model_count.fit(X_train, y_train)

# Make predictions on the test set
y_pred_count_rf = rf_model_count.predict(X_test)

# Print the classification report
print("Classification Report for Random Forest with CountVectorizer:\n", classification_report(y_test, y_pred_count_rf))
```

```
Classification Report for Random Forest with CountVectorizer:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       362
           1       0.87      0.68      0.76       431
           2       0.50      0.83      0.62       999
           3       0.63      0.72      0.67      1039
           4       0.48      0.31      0.37       995

    accuracy                           0.57      3826
   macro avg       0.50      0.51      0.49      3826
weighted avg       0.52      0.57      0.53      3826
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

5.3.2 Random Forest with TFIDFVectorizer

In [ ]:

```python
# Applying the Random Forest on Tf-IDF Vectorizer

# Split the TFIDFVectorizer data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(tfidf_vectors, genre_encoded, test_size=0.2, random_state=42)

# Initialize Random Forest Classifier with reduced number of trees and limited tree depth
rf_model_tfidf = RandomForestClassifier(n_estimators=50, max_depth=10, max_features='sqrt', random_state=42)

# Train the model
rf_model_tfidf.fit(X_train, y_train)

# Make predictions on the test set
y_pred_tfidf_rf = rf_model_tfidf.predict(X_test)

# Print the classification report
print("Classification Report for Random Forest with TFIDFVectorizer:\n", classification_report(y_test, y_pred_tfidf_rf))
```

```
Classification Report for Random Forest with TFIDFVectorizer:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00       362
           1       0.88      0.68      0.77       431
           2       0.52      0.79      0.63       999
           3       0.60      0.76      0.67      1039
           4       0.48      0.32      0.39       995

    accuracy                           0.57      3826
   macro avg       0.50      0.51      0.49      3826
weighted avg       0.52      0.57      0.53      3826
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

5.3.3 Random Forest with Word2Vec

In [ ]:

```python
# Applying the Random Forest on Word2Vec

# Split the Word2Vec mean vectors into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(word2vec_mean_vectors, genre_encoded , test_size=0.2, random_state=42)

# Initialize Random Forest Classifier with reduced number of trees and limited tree depth
rf_model_word2vec = RandomForestClassifier(n_estimators=50, max_depth=10, max_features='sqrt', random_state=42)

# Train the model
rf_model_word2vec.fit(X_train, y_train)

# Make predictions on the test set
y_pred_word2vec_rf = rf_model_word2vec.predict(X_test)

# Print the classification report
print("Classification Report for Random Forest with Word2Vec:\n", classification_report(y_test, y_pred_word2vec_rf))
```

```
Classification Report for Random Forest with Word2Vec:
               precision    recall  f1-score   support

           0       0.50      0.01      0.02       362
           1       0.78      0.66      0.71       431
           2       0.67      0.72      0.69       999
           3       0.66      0.71      0.69      1039
           4       0.45      0.57      0.50       995

    accuracy                           0.60      3826
   macro avg       0.61      0.53      0.52      3826
weighted avg       0.61      0.60      0.58      3826
```

5.3.4 Random Forest with GoogleNews Word2Vec

In [ ]:

```python
# Applying the Random Forest on GoogleNews Word2Vec

# Split the GoogleNews Word2Vec mean vectors into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(google_w2v_mean_vectors, genre_encoded, test_size=0.2, random_state=42)

# Initialize Random Forest Classifier with reduced number of trees and limited tree depth
rf_model_google_w2v = RandomForestClassifier(n_estimators=50, max_depth=10, max_features='sqrt', random_state=42)

# Train the model
rf_model_google_w2v.fit(X_train, y_train)

# Make predictions on the test set
y_pred_google_w2v_rf = rf_model_google_w2v.predict(X_test)

# Print the classification report
print("Classification Report for Random Forest with GoogleNews Word2Vec:\n", classification_report(y_test, y_pred_google_w2v_rf))
```

```
Classification Report for Random Forest with GoogleNews Word2Vec:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       362
           1       0.76      0.63      0.69       431
           2       0.64      0.71      0.67       999
           3       0.66      0.74      0.69      1039
           4       0.44      0.52      0.48       995

    accuracy                           0.59      3826
   macro avg       0.50      0.52      0.51      3826
weighted avg       0.54      0.59      0.57      3826
```

6. Visualizing the Results

6.1 Visualizing the results using bar plots

In [ ]:
```python
from sklearn.metrics import accuracy_score

# Example accuracy calculations (replace the prediction variables with the correct ones from your models)
accuracy_rf_count = accuracy_score(y_test, y_pred_count_rf)
accuracy_rf_tfidf = accuracy_score(y_test, y_pred_tfidf_rf)
accuracy_rf_word2vec = accuracy_score(y_test, y_pred_word2vec_rf)
accuracy_rf_google_w2v = accuracy_score(y_test, y_pred_google_w2v_rf)

accuracy_svc_count = accuracy_score(y_test,y_pred_count_sgd)
accuracy_svc_tfidf = accuracy_score(y_test, y_pred_tfidf_svc)
accuracy_svc_word2vec = accuracy_score(y_test, y_pred_word2vec_sgd_random)
accuracy_svc_google_w2v = accuracy_score(y_test, y_pred_google_w2v_sgd)

accuracy_lr_count = accuracy_score(y_test, y_pred_count)
accuracy_lr_tfidf = accuracy_score(y_test, y_pred_tfidf)
accuracy_lr_word2vec = accuracy_score(y_test, y_pred_word2vec)
accuracy_lr_google_w2v = accuracy_score(y_test, y_pred_google_w2v)
```

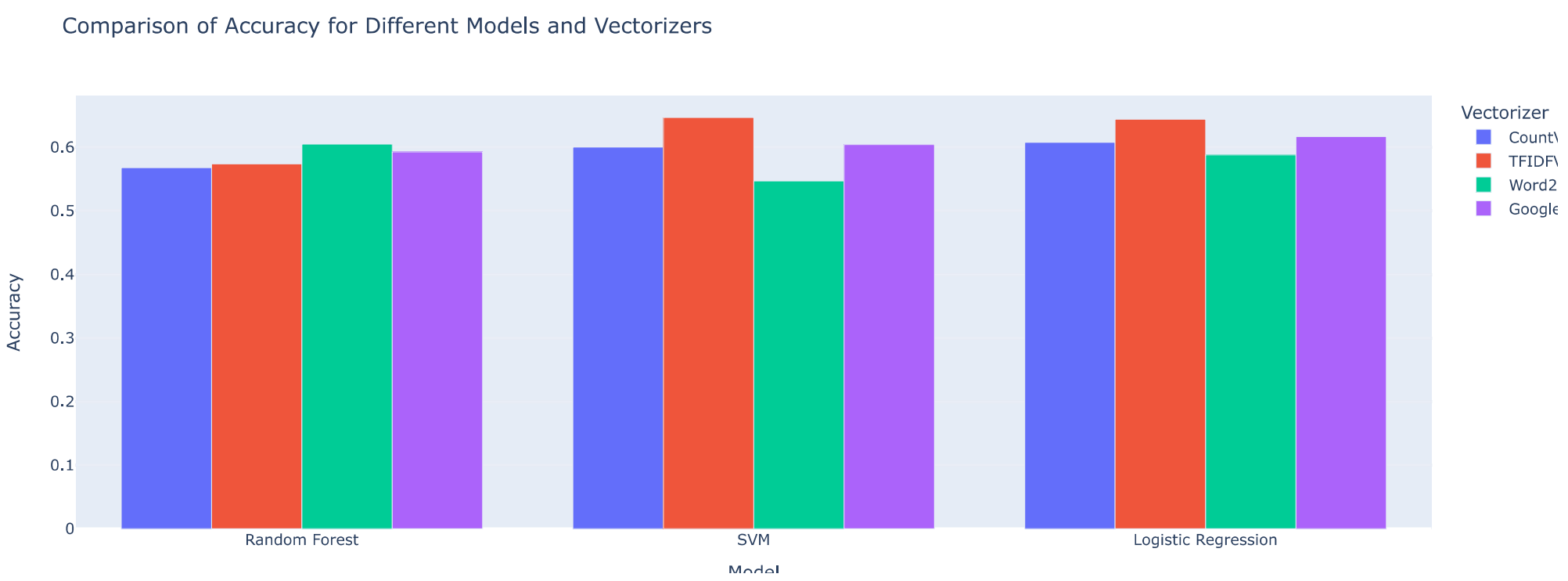
In [ ]:
```python
import plotly.express as px
import pandas as pd
import plotly.io as pio
pio.renderers.default='notebook'

# Prepare data as a list of dictionaries
data = [
    {'Model': 'Random Forest', 'Vectorizer': 'CountVectorizer', 'Accuracy': accuracy_rf_count},
    {'Model': 'Random Forest', 'Vectorizer': 'TFIDFVectorizer', 'Accuracy': accuracy_rf_tfidf},
    {'Model': 'Random Forest', 'Vectorizer': 'Word2Vec', 'Accuracy': accuracy_rf_word2vec},
    {'Model': 'Random Forest', 'Vectorizer': 'GoogleNews Word2Vec', 'Accuracy': accuracy_rf_google_w2v},
    {'Model': 'SVM', 'Vectorizer': 'CountVectorizer', 'Accuracy': accuracy_svc_count},
    {'Model': 'SVM', 'Vectorizer': 'TFIDFVectorizer', 'Accuracy': accuracy_svc_tfidf},
    {'Model': 'SVM', 'Vectorizer': 'Word2Vec', 'Accuracy': accuracy_svc_word2vec},
    {'Model': 'SVM', 'Vectorizer': 'GoogleNews Word2Vec', 'Accuracy': accuracy_svc_google_w2v},
    {'Model': 'Logistic Regression', 'Vectorizer': 'CountVectorizer', 'Accuracy': accuracy_lr_count},
    {'Model': 'Logistic Regression', 'Vectorizer': 'TFIDFVectorizer', 'Accuracy': accuracy_lr_tfidf},
    {'Model': 'Logistic Regression', 'Vectorizer': 'Word2Vec', 'Accuracy': accuracy_lr_word2vec},
    {'Model': 'Logistic Regression', 'Vectorizer': 'GoogleNews Word2Vec', 'Accuracy': accuracy_lr_google_w2v},
]

# Convert to DataFrame
df_plot = pd.DataFrame(data)

# Create a bar plot
fig = px.bar(df_plot, x='Model', y='Accuracy', color='Vectorizer', barmode='group',
             title='Comparison of Accuracy for Different Models and Vectorizers')

# Show the plot
fig.show()
```



6.2 Visualizing the results using sunburst plots

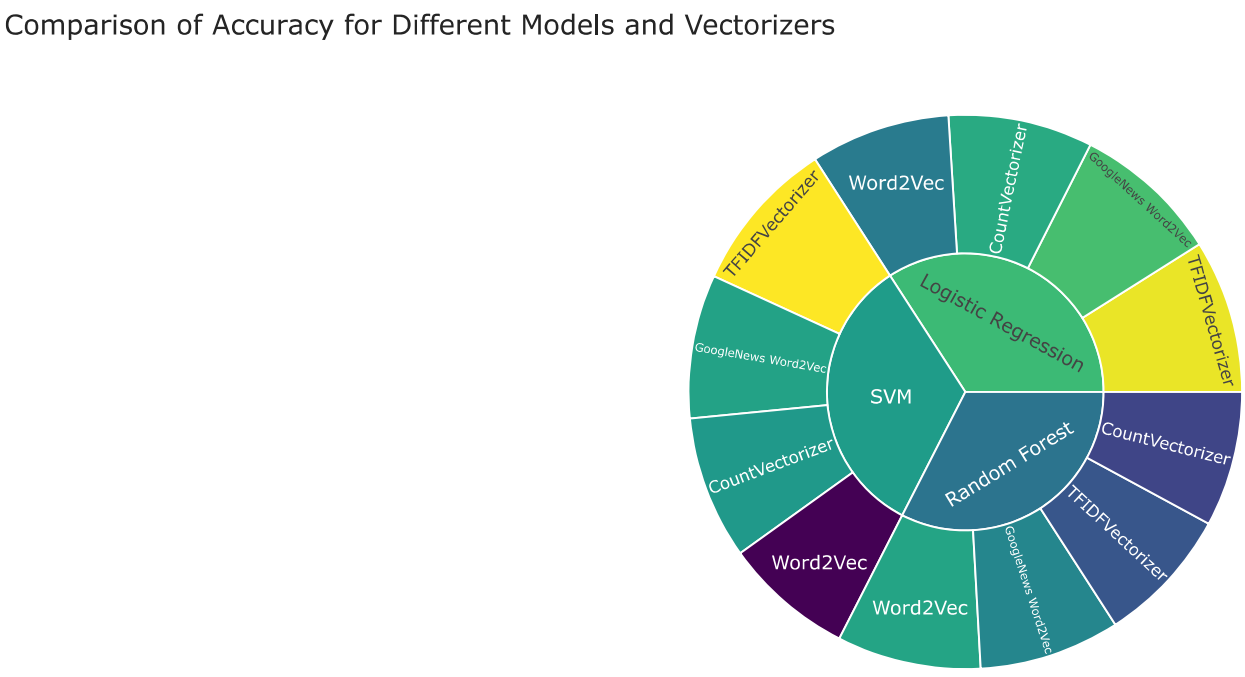
In [ ]:
```python
import plotly.express as px
import pandas as pd
import plotly.io as pio
pio.renderers.default='notebook'
# Prepare data as a list of dictionaries
data = [
    # Random Forest
    {'Model': 'Random Forest', 'Vectorizer': 'CountVectorizer', 'Accuracy': accuracy_rf_count},
    {'Model': 'Random Forest', 'Vectorizer': 'TFIDFVectorizer', 'Accuracy': accuracy_rf_tfidf},
```

```
    {'Model': 'Random Forest', 'Vectorizer': 'Word2Vec', 'Accuracy': accuracy_rf_word2vec},
    {'Model': 'Random Forest', 'Vectorizer': 'GoogleNews Word2Vec', 'Accuracy': accuracy_rf_google_w2v},
    # SVM
    {'Model': 'SVM', 'Vectorizer': 'CountVectorizer', 'Accuracy': accuracy_svc_count},
    {'Model': 'SVM', 'Vectorizer': 'TFIDFVectorizer', 'Accuracy': accuracy_svc_tfidf},
    {'Model': 'SVM', 'Vectorizer': 'Word2Vec', 'Accuracy': accuracy_svc_word2vec},
    {'Model': 'SVM', 'Vectorizer': 'GoogleNews Word2Vec', 'Accuracy': accuracy_svc_google_w2v},
    # Logistic Regression
    {'Model': 'Logistic Regression', 'Vectorizer': 'CountVectorizer', 'Accuracy': accuracy_lr_count},
    {'Model': 'Logistic Regression', 'Vectorizer': 'TFIDFVectorizer', 'Accuracy': accuracy_lr_tfidf},
    {'Model': 'Logistic Regression', 'Vectorizer': 'Word2Vec', 'Accuracy': accuracy_lr_word2vec},
    {'Model': 'Logistic Regression', 'Vectorizer': 'GoogleNews Word2Vec', 'Accuracy': accuracy_lr_google_w2v},
]

# Convert to DataFrame
df_plot = pd.DataFrame(data)

# Create a sunburst plot
fig = px.sunburst(df_plot, path=['Model', 'Vectorizer'], values='Accuracy',
                  title='Comparison of Accuracy for Different Models and Vectorizers',
                  color='Accuracy',  # Color by accuracy
                  color_continuous_scale='viridis',template='simple_white')

# Show the plot
fig.show()
```



7. Conclusion

Conclusion

- The choice of classifier and vectorization technique significantly affects the performance of the classification task.
- The **SGD Classifier with Word2Vec (RandomizedSearchCV)** has the lowest overall accuracy (55%), while the **Random Forest with Word2Vec** has the highest overall accuracy (60%).
- Class-specific performance metrics vary, suggesting that some classes are easier to predict than others.
- Further optimization, feature engineering, or using more advanced models may help improve classification performance.
- Consider conducting more in-depth analysis, such as feature importance and confusion matrix, to gain insights into classifier behavior.

In [ ]:

In [ ]: