databricks finalProject_651_ricardoZambrano

(https://databricks.com)

```
!pip install librosa
```

```
Note: you may need to restart the kernel using dbutils.library.restartPython() to use updated packages.
Collecting librosa
  Using cached librosa-0.10.1-py3-none-any.whl (253 kB)
Requirement already satisfied: scikit-learn>=0.20.0 in /databricks/python3/lib/python3.10/site-packages (from librosa) (1.1.1)
Requirement already satisfied: joblib>=0.14 in /databricks/python3/lib/python3.10/site-packages (from librosa) (1.2.0)
Requirement already satisfied: typing-extensions>=4.1.1 in /databricks/python3/lib/python3.10/site-packages (from librosa) (4.4.0)
Collecting soundfile>=0.12.1
  Using cached soundfile-0.12.1-py2.py3-none-manylinux_2_31_x86_64.whl (1.2 MB)
Requirement already satisfied: decorator>=4.3.0 in /databricks/python3/lib/python3.10/site-packages (from librosa) (5.1.1)
Requirement already satisfied: scipy>=1.2.0 in /databricks/python3/lib/python3.10/site-packages (from librosa) (1.10.0)
Collecting numba>=0.51.0
  Using cached numba-0.58.1-cp310-cp310-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (3.6 MB)
Collecting pooch>=1.0
  Using cached pooch-1.8.0-py3-none-any.whl (62 kB)
Requirement already satisfied: numpy!=1.22.0,!=1.22.1,!=1.22.2,>=1.20.3 in /databricks/python3/lib/python3.10/site-packages (from
librosa) (1.23.5)
Collecting msgpack>=1.0
  Using cached msgpack-1.0.7-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (530 kB)
Collecting lazy-loader>=0.1
  Using cached lazy_loader-0.3-py3-none-any.whl (9.1 kB)
Collecting audioread>=2.1.9
```

```
####################
# Basic Boilerplate #
####################

import os
import sys
import time

# Type Hints (Optional)
from typing import Optional, Tuple, Union, TypeVar, List
#from torch import Tensor
import numpy.typing as npt

import random
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

########################
# NLP and ML Libraries #
########################

from sklearn.datasets import make_classification
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score

import string
import re
#import nltk
#from nltk.tokenize import word_tokenize, TreebankWordTokenizer, wordpunct_tokenize

#################
# Audio Modules #
#################

import librosa

####################
# Big Data Modules #
####################

# Spark NLP
#import sparknlp

# PySpart DataFrame and SQL
from pyspark.sql import SparkSession

from pyspark.sql import Row
from pyspark.sql.functions import concat, col, lit
from pyspark.sql.functions import udf, array, struct
from pyspark.sql.types import StringType
from pyspark.sql.types import StructType
from pyspark.sql.types import StructField
from pyspark.sql.types import IntegerType
from pyspark.sql.types import ArrayType
from pyspark.sql.types import FloatType
from pyspark.sql.functions import pandas_udf, PandasUDFType
from pyspark.sql import functions as F
from pyspark.sql.window import Window
from pyspark.sql.functions import row_number
import pyspark.pandas as ps
from pyspark.sql.functions import when
```

```
# PySpark MLLib
from pyspark.ml.classification import LinearSVC
#from pyspark.ml.classification import MultilayerPerceptronClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.linalg import DenseVector, VectorUDT
from pyspark.mllib.linalg import Vectors
from pyspark.ml.functions import vector_to_array
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import VectorAssembler, StringIndexer, OneHotEncoder


########################################
# Verifying Correct  Working Directory #
########################################

#path_wd = "/mnt/c/Users/rzamb/Documents/UMD/651_Big_Data/finalProjectTest"
#os.chdir(path_wd)
print(os.getcwd())
```

```
/Workspace
```

```
os.listdir('/Volumes/finalproject651/default/common_voice/')
```

```
['accent_groups.csv',
 'accents.csv',
 'common_voice_valid_files.csv',
 'cv-corpus-15.0-2023-09-08-es.tar',
 'cv-corpus-es',
 'fit_model',
 'invalidated.tsv',
 'other.tsv',
 'reported.tsv',
 'result.csv',
 'valid_audio_files.csv',
 'validated.tsv']
```

Global and Helper Functions

```
# To simplyfy the UDF function implememntation and going back and forth from Spark's framework a wrapper
# function was created. It discards the sample rate when loading the waveform. Then inside the function the
# MFCC features are extracted.
# Next the result is padded with zeros in order to get features with the same shape. Finally the numpy array is flattened
# The output is a dense vector
def get_mfcc_features(path:str)->VectorUDT:
    """Load an audio file as a floating point time series.

    Audio will be automatically resampled to the given rate
    (default ``sr=22050``).

    To preserve the native sampling rate of the file, use ``sr=None``.

    Parameters
    ----------
    path : string, int, pathlib.Path, soundfile.SoundFile, audioread object, or file-like object
        path to the input file.

        Any codec supported by `soundfile` or `audioread` will work.

        Any string file paths, or any object implementing Python's
        file interface (e.g. `pathlib.Path`) are supported as `path`.

        If the codec is supported by `soundfile`, then `path` can also be
        an open file descriptor (int) or an existing `soundfile.SoundFile` object.

        Pre-constructed audioread decoders are also supported here, see the example
        below.  This can be used, for example, to force a specific decoder rather
        than relying upon audioread to select one for you.

    Intermediate Results
    --------------------
    ==> librosa.load output <==
    y : np.ndarray [shape=(n,) or (..., n)]
        audio time series. Multi-channel is supported.

    Returns
    -------
    M : np.ndarray [shape=(..., n_mfcc, t)]
        MFCC sequence

    Examples
    --------
    >>> # Load a sample file from common voice
    >>> file_path='/mnt/c/Users/rzamb/Documents/UMD/651_Big_Data/finalProjectTest2/cv-corpus-15.0-delta-2023-09-08-es/cv-corpus-15.0-d
    >>> mfcc = get_mfcc_features(file_path,duration=20)
    >>> mfcc
    DenseVector([-573.9348, -573.9348, -573.9348, -573.8645, -563.9696, -557.1255, -552.7512, -548.8096, -551.2482, -554.7189, ...
    ... 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])
    >>> mfcc.shape
    (57856,)
    """

    # Hard Coded Variables
    win_length = None
    n_fft = 1024
    win_length = None
    hop_length = 512
    n_mels = 128 # Originally 256
    sample_rate = 22050
    n_mfcc = 128 # originally 256
    max_dim = 452

    ### Step 1 ###
    y,_ = librosa.load(path,duration=20) # Discards sample rate wich defaults to 22050
                                         # Limiting the utterances' audio to 20 seconds which is the instructor's recommended length

    ### Step 2 ###
```

```
    mfcc = librosa.feature.mfcc(y=y, sr=sample_rate, n_mfcc=n_mfcc, dct_type=2, norm="ortho")

    ####### Step 3 #######
    # Padding or Cutting #
    ######################
    mfcc_col_len = mfcc.shape[1]
    dim_diff = max_dim - mfcc_col_len
    if dim_diff > 0:
        npad = ((0, 0), (0, dim_diff))
        # Padded_mfcc
        updated_mfcc = np.pad(mfcc, pad_width=npad, mode='constant', constant_values=0)
    elif dim_diff < 0:
        updated_mfcc = mfcc[:,:max_dim]
    elif dim_diff == 0:
        updated_mfcc = mfcc
    else:
        raise ValueError('MFCC features had unexpected shape')

    return DenseVector(updated_mfcc.flatten())
```

```
# Download Common Voice ES dataset
# import urllib
# urllib.request.urlretrieve("https://storage.googleapis.com/common-voice-prod-prod-datasets/cv-corpus-15.0-2023-09-08/cv-corpus-15.0-2023-09-08-es.tar.gz?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gke-prod%40moz-fx-common-voice-prod.iam.gserviceaccount.com%2F20231126%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20231126T004732Z&X-Goog-Expires=43200&X-Goog-SignedHeaders=host&X-Goog-Signature=7533ef6ba904a0e41976c94d3f077a2b82a25beddfae2c39987feb345445b702c18bc8dfbf88b99a1795b44b542fbe6624a747586125f652834e1b54d13a29ba224ebc76b84bd7280e39de63f8e33ac804ef57447b243dcfebd004121dc780a3ec2b738245eecfc566f0b294d8518f42ddd2f84d8ec622da2f8b79cd0c92ff3b361b53e89fa6618f10bf1945b750f926d29a97df175c54486004315c7be0cb0b21ca3e5b69437c950a13b7470f987afa3cf06c18f3476d649865805a52eef21181caa37912c81133d8e1c98f299cd7b3e12a1a911c8ce6b23e838917e7a282eeb21d3e4cf844b9910738056aa1d455b0a60ec44efecc42693c0dc6ab88d6fbaa", "/Volumes/finalproject651/default/common_voice/cv-corpus-15.0-2023-09-08-es.tar")
```

```
# #%sh tar xvf /Volumes/finalproject651/default/common_voice/cv-corpus-15.0-2023-09-08-es.tar -C /Volumes/finalproject651/default/common_voice/cv-corpus-es
# import tarfile
# tar = tarfile.open('/Volumes/finalproject651/default/common_voice/cv-corpus-15.0-2023-09-08-es.tar')
# tar.extractall('/Volumes/finalproject651/default/common_voice/cv-corpus-es')
# tar.close()
```

Loading Common Voice Sub Set with Spark

```
# No ned to start a Spark Session
# Start Spark NLP session
# spark = sparknlp.start()
```

```
# Common voice path
PATH = "/Volumes/finalproject651/default/common_voice/other.tsv"
```

```
# Loading Common Voice
common_voice_df = spark.read.csv(PATH, sep=r'\t',
                         inferSchema=True, header=True)
```

```
common_voice_df.printSchema()
```

```
root
 |-- client_id: string (nullable = true)
 |-- path: string (nullable = true)
 |-- sentence: string (nullable = true)
 |-- up_votes: integer (nullable = true)
 |-- down_votes: integer (nullable = true)
 |-- age: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- accents: string (nullable = true)
 |-- variant: string (nullable = true)
 |-- locale: string (nullable = true)
 |-- segment: string (nullable = true)
```

```
common_voice_df.show(10)
```

```
+--------------------+--------------------+--------------------+--------+----------+--------+------+--------------------+-------+-
-----+-------+
|           client_id|                path|            sentence|up_votes|down_votes|     age|gender|             accents|variant|l
ocale|segment|
+--------------------+--------------------+--------------------+--------+----------+--------+------+--------------------+-------+-
-----+-------+
|f9c44725569f8eeae...|common_voice_es_1...|el indio ya se re...|       0|         1|twenties|  male|Andino-Pacífico: ...|   NULL|
es|   NULL|
|3cc1abdb8e9685355...|common_voice_es_1...|Esta historia tra...|       0|         0|twenties|  male|Rioplatense: Arge...|   NULL|
es|   NULL|
|3cc1abdb8e9685355...|common_voice_es_1...|Ninguno de los pa...|       1|         0|twenties|  male|Rioplatense: Arge...|   NULL|
es|   NULL|
|a97730f86fa90560a...|common_voice_es_1...|En todas las vers...|       1|         0| sixties|  male|España: Sur penin...|   NULL|
es|   NULL|
|c124992069df30bea...|common_voice_es_1...|Luego, como inves...|       1|         0|    NULL|  NULL|                NULL|   NULL|
es|   NULL|
|c124992069df30bea...|common_voice_es_1...|Y es Espartaco qu...|       1|         0|    NULL|  NULL|                NULL|   NULL|
es|   NULL|
|5ac2dcf526f5d61bb...|common_voice_es_1...|Con el apoyo de s...|       1|         0|    NULL|  NULL|                NULL|   NULL|
es|   NULL|
|1681277f5957f1ab3...|common_voice_es_1...|La actividad de l...|       1|         0|   teens|  male|España: Norte pen...|   NULL|
```

```
common_voice_df.count()
```

```
1150345
```

```
# Select columns of interest and dropping rows where the accent is Null
common_voice = common_voice_df.select("path","sentence","accents").where(common_voice_df.accents.isNotNull())
```

```
# Discarding rows where the sentence is not available
common_voice = common_voice.filter(common_voice.sentence.isNotNull())
```

```
# Verifying the dimension of the dataset
print((common_voice.count(), len(common_voice.columns)))
```

```
(932712, 3)
```

```
# Checking number of distinct accents in the dataset
distinct = common_voice.select('accents').distinct().count()
distinct
```

```
111
```

```
common_voice.select('accents').distinct().show(n=distinct)
```

```
+--------------------+
|             accents|
+--------------------+
|América central,G...|
|Rioplatense: Arge...|
|    Ciudad de México|
|Chileno: Chile, C...|
|México,Mexican-Am...|
|     América central|
|México centro,CDM...|
|    Colombian Accent|
|España: Norte pen...|
|España: Noroeste ...|
|   Latino Venezolano|
|España: Norte pen...|
|España: Comunidad...|
|       México,Centro|
|España: Centro-Su...|
|América central,C...|
|Andino-Pacífico: ...|
|España: Norte pen...|
```

Some of these accents need to be grouped. Likewise, some rows need to be discarded because the accent has no information, for example: accent == neutral

```
# Saving the list of acctents. Some accents must be grouped in the same category
# common_voice.select('accents').distinct().write.csv('/Volumes/finalproject651/default/common_voice/accents.csv')
```

```
# Upload a list of the same length to unique accents to create a dict and group accents by grouping
GROUPS_PATH = "/Volumes/finalproject651/default/common_voice/accent_groups.csv"
groups = spark.read.csv(GROUPS_PATH, header=False)
```

```
groups.show()
```

```
+--------------------+
|                 _c0|
+--------------------+
|Rioplatense: Arge...|
|     America Central|
|Espana: Centro-Su...|
|Espana: Norte pen...|
|Chileno: Chile, Cuyo|
|Andino-Pacifico: ...|
|Espanol de Filipinas|
|Caribe: Cuba, Ven...|
|Espana: Islas Can...|
|Espana: Sur penin...|
|              Mexico|
|Andino-Pacifico: ...|
|Caribe: Cuba, Ven...|
```

```
|            Discard|
|    Colombia, Bogota|
|Andino-Pacifico: ...|
|           Guatemala|
```

```
accentMap = [row[0] for row in groups.select('_c0').collect()]
accentsOriginal = [row[0] for row in common_voice.select('accents').distinct().collect()]
```

```
# Now I create a map to update the accent column
accent_map_crosswalk = {k:v for k,v in zip(accentsOriginal,accentMap)}
```

```
# Create a function to update the values on the accent columns. The new groups will be placed on a new column
def group_accents(accents_dict):
    return udf(lambda col: accents_dict.get(col), StringType())

common_voice = common_voice.withColumn("updated_accent", group_accents(accent_map_crosswalk)("accents"))
```

```
common_voice.show(10)
#common_voice.select('updated_accent').distinct().show()
#common_voice.select('updated_accent').distinct().count() == 30
```

```
+--------------------+--------------------+--------------------+--------------------+
|                path|            sentence|             accents|      updated_accent|
+--------------------+--------------------+--------------------+--------------------+
|common_voice_es_1...|el indio ya se re...|Andino-Pacífico: ...|Andino-Pacifico: ...|
|common_voice_es_1...|Esta historia tra...|Rioplatense: Arge...|Rioplatense: Arge...|
|common_voice_es_1...|Ninguno de los pa...|Rioplatense: Arge...|Rioplatense: Arge...|
|common_voice_es_1...|En todas las vers...|España: Sur penin...|Espana: Sur penin...|
|common_voice_es_1...|La actividad de l...|España: Norte pen...|Espana: Norte pen...|
|common_voice_es_1...|Luego, como inves...|España: Norte pen...|Espana: Norte pen...|
|common_voice_es_1...|Decían tener reve...|              México|              Mexico|
|common_voice_es_1...|Era más fácil y m...|     América central|     America Central|
|common_voice_es_1...|Era más fácil y m...|              México|              Mexico|
|common_voice_es_1...|Juega de defensa ...|Andino-Pacífico: ...|Andino-Pacifico: ...|
+--------------------+--------------------+--------------------+--------------------+
only showing top 10 rows
```

```
common_voice =
common_voice.select('path','sentence','updated_accent').withColumn('accents',common_voice.updated_accent).filter(common_voice.up
dated_accent != 'Discard')
```

```
common_voice.count()
```

```
932533
```

```
common_voice = common_voice.drop('updated_accent')
common_voice.show(10)
```

```
+--------------------+--------------------+--------------------+
|                path|            sentence|             accents|
+--------------------+--------------------+--------------------+
|common_voice_es_1...|el indio ya se re...|Andino-Pacifico: ...|
```

```
|common_voice_es_1...|Esta historia tra...|Rioplatense: Arge...|
|common_voice_es_1...|Ninguno de los pa...|Rioplatense: Arge...|
|common_voice_es_1...|En todas las vers...|Espana: Sur penin...|
|common_voice_es_1...|La actividad de l...|Espana: Norte pen...|
|common_voice_es_1...|Luego, como inves...|Espana: Norte pen...|
|common_voice_es_1...|Decían tener reve...|              Mexico|
|common_voice_es_1...|Era más fácil y m...|     America Central|
|common_voice_es_1...|Era más fácil y m...|              Mexico|
|common_voice_es_1...|Juega de defensa ...|Andino-Pacifico: ...|
+--------------------+--------------------+--------------------+
only showing top 10 rows
```

```
common_voice = common_voice.sample(withReplacement=False, fraction=0.4, seed=3)
```

```
common_voice.count()
```

```
373087
```

```
# Setting up the path to the folder where audio files are
AUDIO_FILES_FOLDER_PATH = "/Volumes/finalproject651/default/common_voice/cv-corpus-es/cv-corpus-15.0-2023-09-08/es/clips/"
```

```
# The next step is encoding the accent column. To do this we need to extract the unique values
common_voice.select('accents').distinct().show()
```

```
+--------------------+
|             accents|
+--------------------+
|Rioplatense: Arge...|
|Espana: Sur penin...|
|Espana: Noroeste ...|
|      Francoparlante|
|Mexico, Ciudad de...|
|    Mexican-American|
|Chileno: Chile, Cuyo|
|Caribe: Cuba, Ven...|
|           Lima-Peru|
|              Mexico|
|      Mexico, Centro|
|           Guatemala|
|Espana: Este peni...|
|Espana: Centro-Su...|
|Espanol como segu...|
|     America Central|
|Espana: Norte pen...|
|Espana: Islas Can...|
```

```
accent_class = [item[0] for item in common_voice.select('accents').distinct().collect()]
accent_class
```

```
['Rioplatense: Argentina, Uruguay, este de Bolivia, Paraguay',
 'Espana: Sur peninsular (Andalucia, Extremadura, Murcia)',
 'Chileno: Chile, Cuyo',
 'Caribe: Cuba, Venezuela, Puerto Rico, Republica Dominicana, Panama, Colombia caribena, Mexico caribeno, Costa del golfo de Mexic
o',
 'Mexico',
 'Espanol de Filipinas',
 'Espana: Centro-Sur peninsular (Madrid, Toledo, Castilla-La Mancha)',
 'America Central',
```

```
 'Espana: Norte peninsular (Asturias, Castilla y Leon, Cantabria, Pais Vasco, Navarra, Aragon, La Rioja, Guadalajara, Cuenca)',
 'Espana: Islas Canarias',
 'Andino-Pacifico: Colombia, Peru, Ecuador, oeste de Bolivia y Venezuela andina',
 'Colombia, Bogota',
 'Espana: Noroeste Peninsular,Barcelona',
 'Francoparlante',
 'Mexico, Ciudad de Mexico',
 'Mexican-American',
 'Lima-Peru',
 'Mexico, Centro',
 'Guatemala',
```

```
# We need to create a dict to encode accent classes and another to decode accent classes
accents_encode = {}
for i,accent in enumerate(accent_class):
    accents_encode[accent] = i

accents_decode = {}
for i,accent in enumerate(accent_class):
    accents_decode[i] = accent
```

```
accents_decode
```

```
{0: 'Rioplatense: Argentina, Uruguay, este de Bolivia, Paraguay',
 1: 'Espana: Sur peninsular (Andalucia, Extremadura, Murcia)',
 2: 'Chileno: Chile, Cuyo',
 3: 'Caribe: Cuba, Venezuela, Puerto Rico, Republica Dominicana, Panama, Colombia caribena, Mexico caribeno, Costa del golfo de Me
xico',
 4: 'Mexico',
 5: 'Espanol de Filipinas',
 6: 'Espana: Centro-Sur peninsular (Madrid, Toledo, Castilla-La Mancha)',
 7: 'America Central',
 8: 'Espana: Norte peninsular (Asturias, Castilla y Leon, Cantabria, Pais Vasco, Navarra, Aragon, La Rioja, Guadalajara, Cuenca)',
 9: 'Espana: Islas Canarias',
 10: 'Andino-Pacifico: Colombia, Peru, Ecuador, oeste de Bolivia y Venezuela andina',
 11: 'Colombia, Bogota',
 12: 'Espana: Noroeste Peninsular,Barcelona',
 13: 'Francoparlante',
 14: 'Mexico, Ciudad de Mexico',
 15: 'Mexican-American',
 16: 'Lima-Peru',
 17: 'Mexico, Centro',
 18: 'Guatemala',
 19: 'Espana: Este peninsular, Comunidad Valenciana',
```

```
# Create a function to be called with mapping from a dict
def translate(accents_encode):
    return udf(lambda col: accents_encode.get(col), IntegerType())

common_voice = common_voice.withColumn("encoded_accent", translate(accents_encode)("accents"))
```

```
common_voice.show(10)
```

```
+--------------------+--------------------+--------------------+--------------+
|                path|            sentence|             accents|encoded_accent|
+--------------------+--------------------+--------------------+--------------+
|common_voice_es_1...|el indio ya se re...|Andino-Pacifico: ...|            10|
|common_voice_es_1...|La actividad de l...|Espana: Norte pen...|             8|
|common_voice_es_1...|Luego, como inves...|Espana: Norte pen...|             8|
|common_voice_es_1...|Con María Álvarez...|Andino-Pacifico: ...|            10|
|common_voice_es_1...|Ocupa toda una ma...|Andino-Pacifico: ...|            10|
```

```
|common_voice_es_1...|Vivió en Colima, ...|     America Central|             7|
|common_voice_es_1...|El nombre del tea...|              Mexico|             4|
|common_voice_es_1...|Su espiritualidad...|     America Central|             7|
|common_voice_es_1...|Habita en Asia, E...|              Mexico|             4|
|common_voice_es_1...|El mal tiempo fre...|              Mexico|             4|
+--------------------+--------------------+--------------------+--------------+
only showing top 10 rows
```

```
# Creating a new column with the full path to the audio files
common_voice = common_voice.withColumn("full_path", concat(lit(AUDIO_FILES_FOLDER_PATH),col("path")))
```

```
common_voice.show(10)
```

```
+--------------------+--------------------+--------------------+--------------+--------------------+
|                path|            sentence|             accents|encoded_accent|           full_path|
+--------------------+--------------------+--------------------+--------------+--------------------+
|common_voice_es_2...|A partir de aquí,...|Chileno: Chile, Cuyo|             2|/Volumes/finalpro...|
|common_voice_es_2...|La sede del conda...|Chileno: Chile, Cuyo|             2|/Volumes/finalpro...|
|common_voice_es_2...|Creyendo que habí...|Chileno: Chile, Cuyo|             2|/Volumes/finalpro...|
|common_voice_es_2...|Empezó a asistir ...|Chileno: Chile, Cuyo|             2|/Volumes/finalpro...|
|common_voice_es_2...|La ecuación a tie...|Caribe: Cuba, Ven...|             3|/Volumes/finalpro...|
|common_voice_es_2...|Los caparazones d...|     America Central|             7|/Volumes/finalpro...|
|common_voice_es_2...|Se encuentra prec...|     America Central|             7|/Volumes/finalpro...|
|common_voice_es_2...|En el futuro lo t...|     America Central|             7|/Volumes/finalpro...|
|common_voice_es_2...|El sistema contin...|     America Central|             7|/Volumes/finalpro...|
|common_voice_es_2...|Está considerado ...|     America Central|             7|/Volumes/finalpro...|
+--------------------+--------------------+--------------------+--------------+--------------------+
only showing top 10 rows
```

Filtering to Files Available in AWS S3 Bucket

```
#### Check With Audio Files are in Folder
# files_in_folder = os.listdir(AUDIO_FILES_FOLDER_PATH)
```

```
Command skipped
```

```
# len(files_in_folder)
```

```
Command skipped
```

```
# Creating a dummy column to pinpoint files present in the folder
# common_voice = common_voice.withColumn("in_folder_tree", when(col('path').isin(files_in_folder), 1).otherwise(0)) # Too
cumbersome
```

```
Command skipped
```

```
int(os.path.exists("/Volumes/finalproject651/default/common_voice/cv-corpus-es/cv-corpus-15.0-2023-09-
08/es/clips/common_voice_es_18306544.mp3"))
```

```
Command skipped
```

```
# Creating a dummy column to pinpoint files present in the folder with a more efficient approach. Should check 188k times
instead of 188ktimes in a list of 765k
def in_s3(s3_path):
    return udf(lambda col: int(os.path.exists(col)), IntegerType())

common_voice = common_voice.withColumn("in_folder_tree", in_s3(col("full_path"))("full_path"))
```

Command skipped

```
common_voice.printSchema()
```

Command skipped

```
# common_voice = common_voice.filter(common_voice.in_folder_tree == 1) # Aproach 1. Too cumbersome.
common_voice = common_voice.filter(common_voice.in_folder_tree == 1) # Approach 2
```

Command skipped

```
common_voice.write.csv('/Volumes/finalproject651/default/common_voice/common_voice_valid_files.csv')
```

Command skipped

```
# common_voice.count()
```

Command skipped

```
# common_voice.show(10)
```

Command skipped

Getting MFCC Features

```
PATH_VALID_FILES_DF = '/Volumes/finalproject651/default/common_voice/valid_audio_files.csv'
```

```
list_of_cols=[StructField("file_name",StringType(),True),
              StructField("sentence",StringType(),True),
              StructField("accent",StringType(),True),
              StructField("accent_encoded",IntegerType(),True),
              StructField("full_path",StringType(),True),
              StructField("in_folder_tree",StringType(),True)]
schema=StructType(list_of_cols)
```

```
common_voice_ml = spark.read.csv(PATH_VALID_FILES_DF, sep=',',
                         schema=schema, header=True)
```

Command skipped

```
common_voice_ml.printSchema()
```

Command skipped

```
common_voice_ml.show(10)
```

Command skipped

```
# common_voice_ml.select("full_path").where(common_voice_ml.full_path=='9').show()
common_voice_ml = common_voice_ml.filter(common_voice_ml.full_path != '2')
```

Command skipped

```
common_voice_ml.count()
```

Command skipped

```
# From global functions we are going to regiser the function as an UDF with spark
get_mfcc_features_udf = udf(lambda string: get_mfcc_features(string), VectorUDT())
```

Command skipped

```
common_voice_ml = common_voice_ml.withColumn("mfcc_features",get_mfcc_features_udf(common_voice_ml.full_path))
```

Command skipped

```
common_voice_ml.printSchema()
```

Command skipped

```
# common_voice_ml.write.csv('/Volumes/finalproject651/default/common_voice/common_voice_mfcc.csv')
```

Command skipped

```
# start_time_load = time.time()
# common_voice.show(10)
# print(time.time()-start_time_load)
```

Command skipped

```
# start_time_load = time.time()
# common_voice.show(10)
# print(time.time()-start_time_load)
```

Command skipped

Train-Test Split

```
# Next we split the dataset into train and test sets
common_voice_train, common_voice_test = common_voice_ml.randomSplit(weights=[0.7,0.3], seed=42)
```

Command skipped

```
# start_time_load = time.time()
# common_voice_train.show(10)
# print(time.time()-start_time_load)
```

Command skipped

```
# start_time_load = time.time()
# common_voice_test.show(10)
# print(time.time()-start_time_load)
```

Command skipped

```
# common_voice_train.count()
```

Command skipped

```
# common_voice_test.count()
```

Command skipped

Model

```
log_reg = LogisticRegression(
    featuresCol='mfcc_features',
    labelCol='accent_encoded',
    maxIter=10,
    regParam=0.3,
    elasticNetParam=0.8
)
```

Command skipped

```
# Fitting the model on training data
fit_model = log_reg.fit(common_voice_train)
```

Command skipped

```
# Storing the results on test data
results = fit_model.transform(common_voice_test)
```

Command skipped

```
# Print the coefficients and intercept for multinomial logistic regression
print("Coefficients: \n" + str(fit_model.coefficientMatrix))
print("Intercept: " + str(fit_model.interceptVector))

trainingSummary = fit_model.summary

# Obtain the objective per iteration
objectiveHistory = trainingSummary.objectiveHistory
print("objectiveHistory:")
for objective in objectiveHistory:
    print(objective)

# for multiclass, we can inspect metrics on a per-label basis
print("False positive rate by label:")
for i, rate in enumerate(trainingSummary.falsePositiveRateByLabel):
    print("label %d: %s" % (i, rate))

print("True positive rate by label:")
for i, rate in enumerate(trainingSummary.truePositiveRateByLabel):
    print("label %d: %s" % (i, rate))

print("Precision by label:")
for i, prec in enumerate(trainingSummary.precisionByLabel):
    print("label %d: %s" % (i, prec))

print("Recall by label:")
for i, rec in enumerate(trainingSummary.recallByLabel):
    print("label %d: %s" % (i, rec))

print("F-measure by label:")
for i, f in enumerate(trainingSummary.fMeasureByLabel()):
    print("label %d: %s" % (i, f))

accuracy = trainingSummary.accuracy
falsePositiveRate = trainingSummary.weightedFalsePositiveRate
truePositiveRate = trainingSummary.weightedTruePositiveRate
fMeasure = trainingSummary.weightedFMeasure()
precision = trainingSummary.weightedPrecision
recall = trainingSummary.weightedRecall
print("Accuracy: %s\nFPR: %s\nTPR: %s\nF-measure: %s\nPrecision: %s\nRecall: %s" % (accuracy, falsePositiveRate,
truePositiveRate, fMeasure, precision, recall))
```

```
Command skipped
```

```
accuracy = trainingSummary.accuracy
falsePositiveRate = trainingSummary.weightedFalsePositiveRate
truePositiveRate = trainingSummary.weightedTruePositiveRate
fMeasure = trainingSummary.weightedFMeasure()
precision = trainingSummary.weightedPrecision
recall = trainingSummary.weightedRecall
print("Accuracy: %s\nFPR: %s\nTPR: %s\nF-measure: %s\nPrecision: %s\nRecall: %s" % (accuracy, falsePositiveRate,
truePositiveRate, fMeasure, precision, recall))
```

```
Command skipped
```

```
results.select("accent","accent_encoded","prediction").show(25)
```

```
Command skipped
```

```
trainingSummary.accuracy
```

```
Command skipped
```

```
results.show(25)
```

Command skipped

Try Number 2

```
PATH_VALID_FILES_DF = '/Volumes/finalproject651/default/common_voice/valid_audio_files.csv'

list_of_cols=[StructField("file_name",StringType(),True),
              StructField("sentence",StringType(),True),
                StructField("accent",StringType(),True),
              StructField("accent_encoded",IntegerType(),True),
              StructField("full_path",StringType(),True),
              StructField("in_folder_tree",StringType(),True)]
schema=StructType(list_of_cols)

common_voice_ml = spark.read.csv(PATH_VALID_FILES_DF, sep=',',schema=schema, header=True)

common_voice_ml = common_voice_ml.filter(common_voice_ml.full_path != '2')

# From global functions we are going to regiser the function as an UDF with spark
get_mfcc_features_udf = udf(lambda string: get_mfcc_features(string), VectorUDT())

common_voice_ml = common_voice_ml.withColumn("mfcc_features",get_mfcc_features_udf(common_voice_ml.full_path))

# Next we split the dataset into train and test sets
common_voice_train, common_voice_test = common_voice_ml.randomSplit(weights=[0.7,0.3], seed=42)

log_reg = LogisticRegression(
    featuresCol='mfcc_features',
    labelCol='accent_encoded',
    maxIter=10,
    regParam=0.3,
    elasticNetParam=0.8
)

# Fitting the model on training data
fit_model = log_reg.fit(common_voice_train)
```

```
PATH_TO_MODEL = "/Volumes/finalproject651/default/common_voice/"

fit_model.save(PATH_TO_MODEL + "fit_model")
```

Command skipped

```
trainingSummary = fit_model.summary
```

```
# Storing the results on test data
train_preds = fit_model.transform(common_voice_train)
```

```
train_preds.printSchema()
```

```
root
 |-- file_name: string (nullable = true)
```

```
 |-- sentence: string (nullable = true)
 |-- accent: string (nullable = true)
 |-- accent_encoded: integer (nullable = true)
 |-- full_path: string (nullable = true)
 |-- in_folder_tree: string (nullable = true)
 |-- mfcc_features: vector (nullable = true)
 |-- rawPrediction: vector (nullable = true)
 |-- probability: vector (nullable = true)
 |-- prediction: double (nullable = false)
```

```
train_preds.select("accent_encoded","prediction").write.csv('/Volumes/finalproject651/default/common_voice/train_preds.csv')
```

```
# Storing the results on test data
results = fit_model.transform(common_voice_test)
```

```
results.printSchema()
```

```
root
 |-- file_name: string (nullable = true)
 |-- sentence: string (nullable = true)
 |-- accent: string (nullable = true)
 |-- accent_encoded: integer (nullable = true)
 |-- full_path: string (nullable = true)
 |-- in_folder_tree: string (nullable = true)
 |-- mfcc_features: vector (nullable = true)
 |-- rawPrediction: vector (nullable = true)
 |-- probability: vector (nullable = true)
 |-- prediction: double (nullable = false)
```

```
results.select("accent_encoded","prediction").write.csv('/Volumes/finalproject651/default/common_voice/result2.csv')
```

```
trainingSummary = fit_model.summary
```

```
Command skipped
```

```
trainingSummary.accuracy
#accuracy = trainingSummary.accuracy
#print(accuracy)
```

```
Command skipped
```

```
falsePositiveRate = trainingSummary.weightedFalsePositiveRate
truePositiveRate = trainingSummary.weightedTruePositiveRate
fMeasure = trainingSummary.weightedFMeasure()
precision = trainingSummary.weightedPrecision
recall = trainingSummary.weightedRecall
```

```
Command skipped
```

```
AttributeError: 'LogisticRegressionModel' object has no attribute 'printSchema'
```