MLPClassifier

March 12, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
from google.colab import drive
from sklearn.feature_extraction import text
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
import random
import time
import re
import string
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import SelectKBest, chi2,␣
 ↪f_classif,mutual_info_classif,f_regression
from sklearn.preprocessing import Normalizer
from sklearn import model_selection
from sklearn import svm
import nltk
from nltk import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk.tokenize.treebank import TreebankWordDetokenizer
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
nltk.download('omw-1.4')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

```
nltk.download('wordnet')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data…
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data…
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[1]: True

[2]:
```
#import the data
drive.mount('/content/gdrive/', force_remount=True)

train_data_initial = pd.read_csv('/content/gdrive/MyDrive/ecse551-mp2/train.
 ↪csv')
test_data = pd.read_csv('/content/gdrive/MyDrive/ecse551-mp2/test.csv')

print('shape train:',train_data_initial.shape)
print('shape test:',test_data.shape)
```

```
Mounted at /content/gdrive/
shape train: (718, 2)
shape test: (279, 2)
```

[3]:
```
def shuffle_data(df):
    random.seed(0)  # Use a fixed seed for the random number generator
    df = df.sample(frac=1, random_state=0).reset_index(drop=True)
    return df
```

[4]:
```
#function for creating the test csv file to upload to kaggle
def create_test_csv(data, outfile_name):
  rawdata= {'subreddit':data}
  csv = pd.DataFrame(rawdata, columns = ['subreddit'])
  csv.to_csv(outfile_name,index=True, header=True)
```

```
  print ("File saved.")
```

[5]:
```
#shuffle the data and split the features from the label
train_data = shuffle_data(train_data_initial)

train_x = train_data["body"]
train_y = train_data["subreddit"]
test_x = test_data["body"]
```

[6]:
```
#remove punctuation
def remove_punctuation(text):
   translator = str.maketrans('', '', string.punctuation)
   text = text.translate(translator)
   return text
```

[7]:
```
#remove numeric values,lowercase words
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'\d+', '', text)
    return text
```

[9]:
```
def print_best_params(grid):
  bestParameters = grid.best_estimator_.get_params()
  for paramName in sorted(bestParameters.keys()):
    print("\t%s: %r" % (paramName, bestParameters[paramName]))
```

[10]:
```
#create a dictionary of stop words
stop_words_nltk = set(stopwords.words('english'))
stop_words_sklearn = text.ENGLISH_STOP_WORDS
stop_words_library = stop_words_sklearn.union(stop_words_nltk)
```

[11]:
```
#stemmer lemmatizer
def get_wordnet_pos(word):
    """Map POS tag to first character lemmatize() accepts"""
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}
    return tag_dict.get(tag, wordnet.NOUN)

class LemmaTokenizer_Pos:
     def __init__(self):
       self.wnl = WordNetLemmatizer()
     def __call__(self, doc):
       return [self.wnl.lemmatize(t,pos =get_wordnet_pos(t)) for t in␣
 ↪word_tokenize(doc) if t.isalpha()]
```

```
class LemmaTokenizer:
     def __init__(self):
       self.wnl = WordNetLemmatizer()
     def __call__(self, doc):
       return [self.wnl.lemmatize(t,pos ="v") for t in word_tokenize(doc) if t.
 ↪isalpha()]

class LemmaTokenizer_word:
     def __init__(self):
       self.wnl = WordNetLemmatizer()
     def __call__(self, doc):
       return [self.wnl.lemmatize(t,pos ="v") for t in word_tokenize(doc) ]


class StemTokenizer:
     def __init__(self):
       self.wnl =PorterStemmer()
     def __call__(self, doc):
       return [self.wnl.stem(t) for t in word_tokenize(doc) if t.isalpha()]
```

[12]:
```
stop_words_custom = [
    # All pronouns and associated words
    "i","i'll","i'd","i'm","i've","ive","me","myself","you",
    "you'll",
    "you'd",
    "you're",
    "you've",
    "yourself",
    "he",
    "he'll",
    "he'd",
    "he's",
    "him",
    "she",
    "she'll",
    "she'd",
    "she's",
    "her",
    "it",
    "it'll",
    "it'd",
    "it's",
    "itself",
    "oneself",
    "we",
    "we'll",
```

```
    "we'd",
    "we're",
    "we've",
    "us",
    "ourselves",
    "they",
    "they'll",
    "they'd",
    "they're",
    "they've",
    "them",
    "themselves",
    "everyone",
    "everyone's",
    "everybody",
    "everybody's",
    "someone",
    "someone's",
    "somebody",
    "somebody's",
    "nobody",
    "nobody's",
    "anyone",
    "anyone's",
    "everything",
    "everything's",
    "something",
    "something's",
    "nothing",
    "nothing's",
    "anything",
    "anything's",
    # All determiners and associated words
    "a",
    "an",
    "the",
    "this",
    "that",
    "that's",
    "these",
    "those",
    "my",
    #"mine",   #Omitted since mine can refer to something else
    "your",
    "yours",
    "his",
    "hers",
```

```
    "its",
    "our",
    "ours",
    "own",
    "their",
    "theirs",
    "few",
    "much",
    "many",
    "lot",
    "lots",
    "some",
    "any",
    "enough",
    "all",
    "both",
    "half",
    "either",
    "neither",
    "each",
    "every",
    "certain",
    "other",
    "another",
    "such",
    "several",
    "multiple",
    # "what",    #Dealt with later on
    "rather",
    "quite",
    # All prepositions
    "aboard",
    "about",
    "above",
    "across",
    "after",
    "against",
    "along",
    "amid",
    "amidst",
    "among",
    "amongst",
    "anti",
    "around",
    "as",
    "at",
    "away",
```

```
    "before",
    "behind",
    "below",
    "beneath",
    "beside",
    "besides",
    "between",
    "beyond",
    "but",
    "by",
    "concerning",
    "considering",
    "despite",
    "down",
    "during",
    "except",
    "excepting",
    "excluding",
    "far",
    "following",
    "for",
    "from",
    "here",
    "here's",
    "in",
    "inside",
    "into",
    "left",
    "like",
    "minus",
    "near",
    "of",
    "off",
    "on",
    "onto",
    "opposite",
    "out",
    "outside",
    "over",
    "past",
    "per",
    "plus",
    "regarding",
    "right",
    #"round",   #Omitted
    #"save",    #Omitted
    "since",
```

```
    "than",
    "there",
    "there's",
    "through",
    "to",
    "toward",
    "towards",
    "under",
    "underneath",
    "unlike",
    "until",
    "up",
    "upon",
    "versus",
    "via",
    "with",
    "within",
    "without",
    # Irrelevant verbs
    "may",
    "might",
    "will",
    "won't",
    "would",
    "wouldn't",
    "can",
    "can't",
    "cannot",
    "could",
    "couldn't",
    "should",
    "shouldn't",
    "must",
    "must've",
    "be",
    "being",
    "been",
    "am",
    "are",
    "aren't",
    "ain't",
    "is",
    "isn't",
    "was",
    "wasn't",
    "were",
    "weren't",
```

```
    "do",
    "doing",
    "don't",
    "does",
    "doesn't",
    "did",
    "didn't",
    "done",
    "have",
    "haven't",
    "having",
    "has",
    "hasn't",
    "had",
    "hadn't",
    "get",
    "getting",
    "gets",
    "got",
    "gotten",
    "go",
    "going",
    "gonna",
    "goes",
    "went",
    "gone",
    "make",
    "making",
    "makes",
    "made",
    "take",
    "taking",
    "takes",
    "took",
    "taken",
    "need",
    "needing",
    "needs",
    "needed",
    "use",
    "using",
    "uses",
    "used",
    "want",
    "wanna",
    "wanting",
    "wants",
```

```
    "let",
    "lets",
    "letting",
    "let's",
    "suppose",
    "supposing",
    "supposes",
    "supposed",
    "seem",
    "seeming",
    "seems",
    "seemed",
    "say",
    "saying",
    "says",
    "said",
    "know",
    "knowing",
    "knows",
    "knew",
    "known",
    "look",
    "looking",
    "looked",
    "think",
    "thinking",
    "thinks",
    "thought",
    "feel",
    "feels",
    "felt",
    "based",
    "put",
    "puts",
    #"wanted"   #Omitted since the advective is relevant
    # Question words and associated words
    "who",
    "who's",
    "who've",
    "who'd",
    "whoever",
    "whoever's",
    "whom",
    "whomever",
    "whomever's",
    "whose",
    "whosever",
```

```
    "whosever's",
    "when",
    "whenever",
    "which",
    "whichever",
    "where",
    "where's",
    "where'd",
    "wherever",
    "why",
    "why's",
    "why'd",
    "whyever",
    "what",
    "what's",
    "whatever",
    "whence",
    "how",
    "how's",
    "how'd",
    "however",
    "whether",
    "whatsoever",
    # Connector words and irrelevant adverbs
    "and",
    "or",
    "not",
    "because",
    "also",
    "always",
    "never",
    "only",
    "really",
    "very",
    "greatly",
    "extremely",
    "somewhat",
    "no",
    "nope",
    "nah",
    "yes",
    "yep",
    "yeh",
    "yeah",
    "maybe",
    "perhaps",
    "more",
```

```
    "most",
    "less",
    "least",
    "good",
    "great",
    "well",
    "better",
    "best",
    "bad",
    "worse",
    "worst",
    "too",
    "thru",
    "though",
    "although",
    "yet",
    "already",
    "then",
    "even",
    "now",
    "sometimes",
    "still",
    "together",
    "altogether",
    "entirely",
    "fully",
    "entire",
    "whole",
    "completely",
    "utterly",
    "seemingly",
    "apparently",
    "clearly",
    "obviously",
    "actually",
    "actual",
    "usually",
    "usual",
    "literally",
    "honestly",
    "absolutely",
    "definitely",
    "generally",
    "totally",
    "finally",
    "basically",
    "essentially",
```

```
    "fundamentally",
    "automatically",
    "immediately",
    "necessarily",
    "primarily",
    "normally",
    "perfectly",
    "constantly",
    "particularly",
    "eventually",
    "hopefully",
    "mainly",
    "typically",
    "specifically",
    "differently",
    "appropriately",
    "plenty",
    "certainly",
    "unfortunately",
    "ultimately",
    "unlikely",
    "likely",
    "potentially",
    "fortunately",
    "personally",
    "directly",
    "indirectly",
    "nearly",
    "closely",
    "slightly",
    "probably",
    "possibly",
    "especially",
    "frequently",
    "often",
    "oftentimes",
    "seldom",
    "rarely",
    "sure",
    "while",
    "whilst",
    "able",
    "unable",
    "else",
    "ever",
    "once",
    "twice",
```

```
    "thrice",
    "almost",
    "again",
    "instead",
    "next",
    "previous",
    "unless",
    "somehow",
    "anyhow",
    "anywhere",
    "somewhere",
    "everywhere",
    "nowhere",
    "further",
    "anymore",
    "later",
    "ago",
    "ahead",
    "just",
    "same",
    "different",
    "big",
    "small",
    "little",
    "tiny",
    "large",
    "huge",
    "pretty",
    "mostly",
    "anyway",
    "anyways",
    "otherwise",
    "regardless",
    "throughout",
    "additionally",
    "moreover",
    "furthermore",
    "meanwhile",
    "afterwards",
    # Irrelevant nouns
    "thing",
    "thing's",
    "things",
    "stuff",
    "other's",
    "others",
    "another's",
```

```
    "total",
    "",
    "false",
    "none",
    "way",
    "kind",
    # Lettered numbers and order
    "zero",
    "zeros",
    "zeroes",
    "one",
    "ones",
    "two",
    "three",
    "four",
    "five",
    "six",
    "seven",
    "eight",
    "nine",
    "ten",
    "twenty",
    "thirty",
    "forty",
    "fifty",
    "sixty",
    "seventy",
    "eighty",
    "ninety",
    "hundred",
    "hundreds",
    "thousand",
    "thousands",
    "million",
    "millions",
    "first",
    "last",
    "second",
    "third",
    "fourth",
    "fifth",
    "sixth",
    "seventh",
    "eigth",
    "ninth",
    "tenth",
    "firstly",
```

```
    "secondly",
    "thirdly",
    "lastly",
    # Greetings and slang
    "hello",
    "hi",
    "hey",
    "sup",
    "yo",
    "greetings",
    "please",
    "okay",
    "ok",
    "y'all",
    "lol",
    "rofl",
    "thank",
    "thanks",
    "alright",
    "kinda",
    "dont",
    "sorry",
    "idk",
    "tldr",
    "tl",
    "dr",  #This means that dr (doctor) is a bad feature because of tl;dr
    "tbh",
    "dude",
    "tho",
    "aka",
    "plz",
    "pls",
    "bit",
    "don",
    # Miscellaneous
    "www",
    "https",
    "http",
    "com",
    "etc",
    "html",
    "reddit",
    "subreddit",
    "subreddits",
    "comments",
    "reply",
    "replies",
```

```
    "thread",
    "threads",
    "post",
    "posts",
    "website",
    "websites",
    "web site",
    "web sites"]
print('length custom:',len(stop_words_custom))
```

```
length custom: 590
```

[ ]:
```
############################################################
```

[13]:
```
from sklearn.neural_network import MLPClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
import numpy as np


t_start = time.time()

# Define the hyperparameters to search over
parameters = {
   'tfidf__max_df': (0.25, 0.5),
    'clf__hidden_layer_sizes': [(100,)],
    'selecter__k':[2500],
    'tfidf__ngram_range':[(1,1)],
    #'clf__learning_rate':['adaptive'],
    'clf__activation':["relu"],
        'clf__solver':["adam"],
    'clf__max_iter':[2000],
     "tfidf__stop_words": [list(stop_words_custom)],
    'clf__alpha': [0.1]
}

# Define the MLP architecture
mlp = MLPClassifier()
normalizer = Normalizer()
selecter = SelectKBest(chi2)


# Create the pipeline
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ("selecter", selecter),
```

```
    ("normalizer", normalizer),
    ('clf', mlp)
])



# Create the grid search object
grid_search = GridSearchCV(pipeline, parameters, cv=5, verbose=1, n_jobs=-1)

# Fit the grid search to the data
grid_search.fit(train_x, train_y)


t_end = time.time()

elapsed_time = t_end-t_start
accuracy = round(grid_search.best_score_ * 100,3)

print(f"The best accuracy is {accuracy}.")
print(f"The winning parameters are {grid_search.best_params_}")
print(f"Run time: {elapsed_time} seconds")
```

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits

/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:409:
UserWarning: Your stop_words may be inconsistent with your preprocessing.
Tokenizing the stop words generated tokens ['ain', 'aren', 'couldn', 'didn',
'doesn', 'hadn', 'hasn', 'haven', 'isn', 'll', 're', 'shouldn', 'site', 'sites',
've', 'wasn', 'web', 'weren', 'won', 'wouldn'] not in stop_words.
  warnings.warn(

The best accuracy is 94.843.
The winning parameters are {'clf__activation': 'relu', 'clf__alpha': 0.1,
'clf__hidden_layer_sizes': (100,), 'clf__max_iter': 2000, 'clf__solver': 'adam',
'selecter__k': 2500, 'tfidf__max_df': 0.5, 'tfidf__ngram_range': (1, 1),
'tfidf__stop_words': ['i', "i'll", "i'd", "i'm", "i've", 'ive', 'me', 'myself',
'you', "you'll", "you'd", "you're", "you've", 'yourself', 'he', "he'll", "he'd",
"he's", 'him', 'she', "she'll", "she'd", "she's", 'her', 'it', "it'll", "it'd",
"it's", 'itself', 'oneself', 'we', "we'll", "we'd", "we're", "we've", 'us',
'ourselves', 'they', "they'll", "they'd", "they're", "they've", 'them',
'themselves', 'everyone', "everyone's", 'everybody', "everybody's", 'someone',
"someone's", 'somebody', "somebody's", 'nobody', "nobody's", 'anyone',
"anyone's", 'everything', "everything's", 'something', "something's", 'nothing',
"nothing's", 'anything', "anything's", 'a', 'an', 'the', 'this', 'that',
"that's", 'these', 'those', 'my', 'your', 'yours', 'his', 'hers', 'its', 'our',
'ours', 'own', 'their', 'theirs', 'few', 'much', 'many', 'lot', 'lots', 'some',
'any', 'enough', 'all', 'both', 'half', 'either', 'neither', 'each', 'every',
'certain', 'other', 'another', 'such', 'several', 'multiple', 'rather', 'quite',
```

'aboard', 'about', 'above', 'across', 'after', 'against', 'along', 'amid',
'amidst', 'among', 'amongst', 'anti', 'around', 'as', 'at', 'away', 'before',
'behind', 'below', 'beneath', 'beside', 'besides', 'between', 'beyond', 'but',
'by', 'concerning', 'considering', 'despite', 'down', 'during', 'except',
'excepting', 'excluding', 'far', 'following', 'for', 'from', 'here', "here's",
'in', 'inside', 'into', 'left', 'like', 'minus', 'near', 'of', 'off', 'on',
'onto', 'opposite', 'out', 'outside', 'over', 'past', 'per', 'plus',
'regarding', 'right', 'since', 'than', 'there', "there's", 'through', 'to',
'toward', 'towards', 'under', 'underneath', 'unlike', 'until', 'up', 'upon',
'versus', 'via', 'with', 'within', 'without', 'may', 'might', 'will', "won't",
'would', "wouldn't", 'can', "can't", 'cannot', 'could', "couldn't", 'should',
"shouldn't", 'must', "must've", 'be', 'being', 'been', 'am', 'are', "aren't",
"ain't", 'is', "isn't", 'was', "wasn't", 'were', "weren't", 'do', 'doing',
"don't", 'does', "doesn't", 'did', "didn't", 'done', 'have', "haven't",
'having', 'has', "hasn't", 'had', "hadn't", 'get', 'getting', 'gets', 'got',
'gotten', 'go', 'going', 'gonna', 'goes', 'went', 'gone', 'make', 'making',
'makes', 'made', 'take', 'taking', 'takes', 'took', 'taken', 'need', 'needing',
'needs', 'needed', 'use', 'using', 'uses', 'used', 'want', 'wanna', 'wanting',
'wants', 'let', 'lets', 'letting', "let's", 'suppose', 'supposing', 'supposes',
'supposed', 'seem', 'seeming', 'seems', 'seemed', 'say', 'saying', 'says',
'said', 'know', 'knowing', 'knows', 'knew', 'known', 'look', 'looking',
'looked', 'think', 'thinking', 'thinks', 'thought', 'feel', 'feels', 'felt',
'based', 'put', 'puts', 'who', "who's", "who've", "who'd", 'whoever',
"whoever's", 'whom', 'whomever', "whomever's", 'whose', 'whosever',
"whosever's", 'when', 'whenever', 'which', 'whichever', 'where', "where's",
"where'd", 'wherever', 'why', "why's", "why'd", 'whyever', 'what', "what's",
'whatever', 'whence', 'how', "how's", "how'd", 'however', 'whether',
'whatsoever', 'and', 'or', 'not', 'because', 'also', 'always', 'never', 'only',
'really', 'very', 'greatly', 'extremely', 'somewhat', 'no', 'nope', 'nah',
'yes', 'yep', 'yeh', 'yeah', 'maybe', 'perhaps', 'more', 'most', 'less',
'least', 'good', 'great', 'well', 'better', 'best', 'bad', 'worse', 'worst',
'too', 'thru', 'though', 'although', 'yet', 'already', 'then', 'even', 'now',
'sometimes', 'still', 'together', 'altogether', 'entirely', 'fully', 'entire',
'whole', 'completely', 'utterly', 'seemingly', 'apparently', 'clearly',
'obviously', 'actually', 'actual', 'usually', 'usual', 'literally', 'honestly',
'absolutely', 'definitely', 'generally', 'totally', 'finally', 'basically',
'essentially', 'fundamentally', 'automatically', 'immediately', 'necessarily',
'primarily', 'normally', 'perfectly', 'constantly', 'particularly',
'eventually', 'hopefully', 'mainly', 'typically', 'specifically', 'differently',
'appropriately', 'plenty', 'certainly', 'unfortunately', 'ultimately',
'unlikely', 'likely', 'potentially', 'fortunately', 'personally', 'directly',
'indirectly', 'nearly', 'closely', 'slightly', 'probably', 'possibly',
'especially', 'frequently', 'often', 'oftentimes', 'seldom', 'rarely', 'sure',
'while', 'whilst', 'able', 'unable', 'else', 'ever', 'once', 'twice', 'thrice',
'almost', 'again', 'instead', 'next', 'previous', 'unless', 'somehow', 'anyhow',
'anywhere', 'somewhere', 'everywhere', 'nowhere', 'further', 'anymore', 'later',
'ago', 'ahead', 'just', 'same', 'different', 'big', 'small', 'little', 'tiny',
'large', 'huge', 'pretty', 'mostly', 'anyway', 'anyways', 'otherwise',

```
'regardless', 'throughout', 'additionally', 'moreover', 'furthermore',
'meanwhile', 'afterwards', 'thing', "thing's", 'things', 'stuff', "other's",
'others', "another's", 'total', '', 'false', 'none', 'way', 'kind', 'zero',
'zeros', 'zeroes', 'one', 'ones', 'two', 'three', 'four', 'five', 'six',
'seven', 'eight', 'nine', 'ten', 'twenty', 'thirty', 'forty', 'fifty', 'sixty',
'seventy', 'eighty', 'ninety', 'hundred', 'hundreds', 'thousand', 'thousands',
'million', 'millions', 'first', 'last', 'second', 'third', 'fourth', 'fifth',
'sixth', 'seventh', 'eigth', 'ninth', 'tenth', 'firstly', 'secondly', 'thirdly',
'lastly', 'hello', 'hi', 'hey', 'sup', 'yo', 'greetings', 'please', 'okay',
'ok', "y'all", 'lol', 'rofl', 'thank', 'thanks', 'alright', 'kinda', 'dont',
'sorry', 'idk', 'tldr', 'tl', 'dr', 'tbh', 'dude', 'tho', 'aka', 'plz', 'pls',
'bit', 'don', 'www', 'https', 'http', 'com', 'etc', 'html', 'reddit',
'subreddit', 'subreddits', 'comments', 'reply', 'replies', 'thread', 'threads',
'post', 'posts', 'website', 'websites', 'web site', 'web sites']}
Run time: 78.14167332649231 seconds
```

[14]:
```
print(round(grid_search.best_score_ * 100,3))
print(f"Run time: {elapsed_time} seconds")
y_pred = grid_search.predict(test_x)
create_test_csv(y_pred,"CNN_07032023_01.csv")
```

```
94.982
Run time: 36.68663692474365 seconds
File saved.
```

[ ]:
```
################################################################################
```