bullyingdetection.py

Scanned on: 19:7 May 3, 2023 UTC

Overall similarity score

Results found

Total words in text

	Word count
Identical	95
Minor Changes	6
Paraphrased	3
Omitted	0

bullyingdetection.py

Scanned on: 19:7 May 3, 2023 UTC

Results

The results include any sources we have found in your submitted document that includes the following: identical text, minor changed text, paraphrased text.

Sentiment Analysis with Text Mining *https://www.freecodecamp.org/news/sentiment-analysis-with-text-mining/*	6%
NN - Multi-layer Perceptron Classifier (MLPClassifier) - Mich... *https://michael-fuchs-python.netlify.app/2021/02/03/nn-multi-layer-perce...*	3%
Out-of-core classification of text documents — scikit-learn 1... *https://scikit-learn.org/stable/auto_examples/applications/plot_out_of_cor...* plot_out_of_core_classification.html	3%
Confusion Matrix for Multi-Class Classification | Latest Guid... *https://www.analyticsvidhya.com/blog/2021/06/confusion-matrix-for-mult...*	3%
Sentiment Analysis with Text Mining | by Bert Carremans | ... *https://towardsdatascience.com/sentiment-analysis-with-text-mining-13d...*	3%
classification - Training accuracy on Naive Bayes in Python - ... *https://stackoverflow.com/questions/57899609/training-accuracy-on-naiv...*	2%
Top 145 Python Interview Questions for 2023- Great Learning *https://www.mygreatlearning.com/blog/python-interview-questions/*	2%
implementation - How to calculate true positive, true negat... *https://datascience.stackexchange.com/questions/104575/how-to-calculat...*	2%

IDENTICAL

Text that is exactly the same.

MINOR CHANGES

Text that is nearly identical, yet a different form of the word is used. (i.e 'slow' becomes 'slowly')

PARAPHRASED

Text that has similar meaning, yet different words are used to convey the same message.

Unsure about your report?

The results have been found after comparing your submitted text to online sources, open databases and the Copyleaks internal database. If you have any questions or concerns, please feel free to contact us atsupport@copyleaks.com

Click here to learn more about different types of plagiarism

bullyingdetection.py

Scanned on: 19:7 May 3, 2023 UTC

Text Preprocessing in Python | Set - 1 - GeeksforGeeks *https://www.geeksforgeeks.org/text-preprocessing-in-python-set-1/*	2%
Confusion matrix — scikit-learn 1.2.2 documentation *https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusi...* plot_confusion_matrix.html	1%
All About Heatmaps. The Comprehensive Guide | by Shrasht... *https://towardsdatascience.com/all-about-heatmaps-bb7d97f099d7?gi=2...*	1%
Implementing a Random Forest Classification Model in Pyth... *https://medium.com/@hjhuney/implementing-a-random-forest-classificati...*	1%
Generate classification report and confusion matrix in Pyth... *https://www.projectpro.io/recipes/generate-classification-report-and-conf...*	1%
Basic Tweet Preprocessing in Python | by Parthvi Shah | To... *https://towardsdatascience.com/basic-tweet-preprocessing-in-python-efd8...*	1%

bullyingdetection.py

Scanned on: 19:7 May 3, 2023 UTC

Scanned Text

Your text is highlighted according to the plagiarism types that where found, as shown above.

IDENTICAL MINOR CHANGES PARAPHRASED

```
# -*- coding: utf-8 -*-
"""bullyingDetection.ipynb

Automatically generated by Colaboratory.

Original file is located at
https://colab.research.google.com/drive/1yholejiLlUapUBRZE8SB3xuCu_qxkSn7
"""

from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split

"""Importing modules of classification algorithms"""

from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn import tree
from sklearn import svm
from sklearn.neural_network import MLPClassifier

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tokenize import word_tokenize
import string
from nltk.stem import WordNetLemmatizer

import re
import csv
import pandas as pd
import nltk
import numpy as np
from nltk.corpus import stopwords

"""Reading The CSV file and deleting empty rows if present"""

df = pd.read_csv("/content/cyberbullying_tweets.csv",engine="python")
# df.info()
```

```
# df.head()
df.dropna()
df.info()

df['tweet_text']

"""Converting entire text to lowercase"""

def textLowerCase(text):
return text.lower()

"""Removes Unnecessary data from the text"""

def removeUrls(text):
newText = ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", text).split())
return newText

"""Removing Numbers from the text"""

def removeNumbers(text):
result = re.sub(r'd+','',text)
return result

"""Removing punctuations from the text"""

def remove_punctuation(text):
translator = str.maketrans('', '', string.punctuation)
return text.translate(translator)

"""Tokenizing the text"""

def tokenize(text):
text = word_tokenize(text)
return text

nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('omw-1.4')
# print(stopwords)

"""Removing stop words from text"""

stop_words = set(stopwords.words('english'))
def removeStopWords(text):
text = [value for value in text if not value in stop_words]
return text

"""lemmatizing the text"""

lemmatizer = WordNetLemmatizer()
def lemmatize(text):
text = [lemmatizer.lemmatize(token) for token in text]
return text

"""Function to Preprocess the text

"""

def preProcessingText(text):
text = textLowerCase(text)
text = removeUrls(text)
text = removeNumbers(text)
text = remove_punctuation(text)
text = tokenize(text)
text = removeStopWords(text)
```

```
text = lemmatize(text)
text = ' '.join(text)
return text

"""Data Preprocessing"""

preProcessedText = []
for text_data in df['tweet_text']:
ProcessedText = preProcessingText(text_data)
preProcessedText.append(ProcessedText)
df['tweet_text'] = preProcessedText
yComp = np.array(df['cyberbullying_type'])

df['tweet_text']

v = TfidfVectorizer(min_df=10, max_features=10000)
X_comm = v.fit_transform(df['tweet_text'].values.astype('U')).toarray()

df['cyberbullying_type'].value_counts()

"""Splitting data into train and test data sets"""

X_train_comm, X_test_comm, y_train_comm, y_test_comm = train_test_split(X_comm, yComp,
test_size=0.3)

"""Initialzing classification models"""

naiveBayesClassifier = MultinomialNB()
RandomClassifier = RandomForestClassifier(n_estimators=150)

"""Training Random forest classifier"""

RandomClassifier.fit(X_train_comm, y_train_comm)
y_pred_comm = RandomClassifier.predict(X_test_comm)
# print(y_pred_comm)

# inputText = input("Enter text you want: ")
# inputText = v.transform([inputText]).toarray()
# # print(inputText)
# result = RandomClassifier.predict(inputText)
# print(result)

"""Training Naive Bayes classifier"""

naiveBayesClassifier.fit(X_train_comm ,y_train_comm)
y_pred_nb = naiveBayesClassifier.predict(X_test_comm)
# print(y_pred_nb)

# inputText = input("Enter text you want: ")
# inputText = v.transform([inputText]).toarray()
# # print(inputText)
# result = naiveBayesClassifier.predict(inputText)
# print(result)

"""Training Decision Tree classfier"""

dc_tree = tree.DecisionTreeClassifier(max_features=800)
dc_tree.fit(X_train_comm, y_train_comm)
y_pred_dc = dc_tree.predict(X_test_comm)

"""Testing Decision tree classfier"""

inputText = input("Enter text you want: ")
inputText = v.transform([inputText]).toarray()
# print(inputText)
result = dc_tree.predict(inputText)
```

```
print(result)

mlp = MLPClassifier(alpha=1, max_iter=1000)
mlp.fit(X_train_comm, y_train_comm)
y_pred_nnc = mlp.predict(X_test_comm)

"""Testing Multilayer perceptron model"""

inputText = input("Enter text you want: ")
inputText = v.transform([inputText]).toarray()
# print(inputText)
result = mlp.predict(inputText)
print(result)

"""Testing all the models and outputs

"""

def testAllClassfiers(text):
text = v.transform([text]).toarray()
random_res = RandomClassifier.predict(text)
naiveBayesRes =naiveBayesClassifier.predict(text)
dcRes = dc_tree.predict(text)
mlpRes = mlp.predict(text)
print("Result of Random Forest model : ", random_res)
print("Result of Naive Bayes Model : ", naiveBayesRes)
print("Result of the Decision Tress MOdel : ", dcRes)
print("Result of Neural Networks Model : ", mlpRes)

inputText = input("Enter text here :")
testAllClassfiers(inputText)

inputText = input("Enter text here :")
testAllClassfiers(inputText)

inputText = input("Enter text here :")
testAllClassfiers(inputText)

"""Naive Bayes accuracy """

a = confusion_matrix(y_test_comm,y_pred_nb)
print("Confusion Matrix of Naive Bayes classifier")
print(a)

totalCount = sum(sum(a))
# print("Total length of test dataset :{}".format(sum(sum(a))))
truePositiveSum = 0
for i in range(len(a)):
truePositiveSum += a[i][i]
print("Total true positive value count :{}".format(truePositiveSum))
print("Accuracy of Naive Base model is: {}%".format((truePositiveSum/totalCount)*100))

"""Random Forest Accuracy """

yPred_randomClass = RandomClassifier.predict(X_test_comm)
print(yPred_randomClass)

a = confusion_matrix(y_test_comm,yPred_randomClass)
print("Confusion Matrix of Random Forest classifier")
print(a)

truePositiveSum = 0
for i in range(len(a)):
truePositiveSum += a[i][i]
print("Total true positive value count :{}".format(truePositiveSum))
print("Accuracy of Random forest model is: {}%".format((truePositiveSum/totalCount)*100))
```

```
"""Confusion matrix of decision tree classifer and it's accuracy"""

a = confusion_matrix(y_test_comm,y_pred_dc)
print("Confusion Matrix of decision tree classifier")
print(a)

truePositiveSum = 0
for i in range(len(a)):
truePositiveSum += a[i][i]
print("Total true positive value count :{}".format(truePositiveSum))
print("Accuracy of decision tree model is: {}%".format((truePositiveSum/totalCount)*100))

"""Confusion matrix of MLP classifier and it's accuracy"""

a = confusion_matrix(y_test_comm,y_pred_nnc)
print("Confusion Matrix of MLP classifier")
print(a)

truePositiveSum = 0
for i in range(len(a)):
truePositiveSum += a[i][i]
print("Total true positive value count :{}".format(truePositiveSum))
print("Accuracy of MLP classifier model is: {}%".format((truePositiveSum/totalCount)*100))

"""Heat maps of corresponding algorithms"""

def heatMaps(test , pred):
array = confusion_matrix(test, pred)
df_cm = pd.DataFrame(array, range(6), range(6))
plt.figure(figsize=(10, 7))
# sn.set(font_scale=1.4) # for label size
sn.heatmap(df_cm, annot=True, annot_kws={"size": 15}) # font size
plt.show()

"""NaiveBayes classifier heatmap"""

heatMaps(y_test_comm,y_pred_nb)

"""Random forest classifier heatmap"""

heatMaps(y_test_comm,yPred_randomClass)

"""HeatMap of decision tree classifier"""

heatMaps(y_test_comm,y_pred_dc)

"""HeatMap of MLP classifier"""

heatMaps(y_test_comm,y_pred_nnc)
```