code_plgarism

anonymous marking enabled

Submission date: 16-Jul-2023 07:57PM (UTC-0700)
Submission ID: 2132257600
File name: code_plgarism.txt (11.27K)
Word count: 813
Character count: 10421

```
# for input
drive.mount('/content/drive')
#import BASIC LIbraries,module
```

15

```
import numpy as NP
import pandas as PD
import matplotlib.pyplot as plt
import random
from sklearn.model_selection import learning_curve, validation_curve
```

19

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score
#these libraries useful for train ,test and handle large numbers and data sets
train_X=NP.loadtxt('/content/drive/MyDrive/stress_mini_project/Project/Dataset/train_input.csv',delimiter=",")
train_Y=NP.loadtxt('/content/drive/MyDrive/stress_mini_project/Project/Dataset/train_labels.csv')
test_X=NP.loadtxt('/content/drive/MyDrive/stress_mini_project/Project/Dataset/test_input.csv',delimiter=",")
test_Y=NP.loadtxt('/content/drive/MyDrive/stress_mini_project/Project/Dataset/test_labels.csv')
train_X=train_X.reshape(10000,48,48,1)
train_Y=train_Y.reshape(10000,1)
test_X=test_X.reshape(2000,48,48,1)
```

```
test_Y=test_Y.reshape(2000,1)

train_X=train_X/255 #Normalization of pixel data

test_X=test_X/255

PIC_TEST=random.randint(0,9999)

plt.imshow(train_X[PIC_TEST,:],cmap='gray')

plt.show()

PIC_TEST=random.randint(0,1999)

plt.imshow(test_X[PIC_TEST,:],cmap='gray')

plt.show()

from keras.models import Sequential

from keras.layers import Activation,Dropout,Conv2D,Dense

from keras.layers import BatchNormalization

from keras.layers import MaxPooling2D

from keras.layers import Flatten

design=Sequential() #cnn layers adding

design.add(Conv2D(filters=16, kernel_size=(7, 7), padding='same',

input_shape=(48,48,1))) # features extraction

design.add(BatchNormalization())

design.add(Conv2D(filters=16, kernel_size=(7, 7), padding='same'))

design.add(BatchNormalization())

design.add(Activation('relu'))

design.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

design.add(Dropout(.5))
```

```
design.add(Conv2D(filters=32, kernel_size=(5, 5), padding='same'))

design.add(BatchNormalization())

design.add(Conv2D(filters=32, kernel_size=(5, 5), padding='same'))

design.add(BatchNormalization())

design.add(Activation('relu'))

design.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

design.add(Dropout(.5))

design.add(Conv2D(filters=64, kernel_size=(3, 3), padding='same'))

design.add(BatchNormalization())

design.add(Conv2D(filters=64, kernel_size=(3, 3), padding='same'))

design.add(BatchNormalization())

design.add(Activation('relu'))

design.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

design.add(Dropout(.5))

design.add(Conv2D(filters=128, kernel_size=(3, 3), padding='same'))

design.add(BatchNormalization())

design.add(Conv2D(filters=128, kernel_size=(3, 3), padding='same'))

design.add(BatchNormalization())

design.add(Activation('relu'))

design.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

design.add(Dropout(.5))

design.add(Flatten())

design.add(Dense(units=256,activation='relu'))
```

```
design.add(Dense(units=128,activation='relu'))

design.add(Dense(units=1,activation='sigmoid'))

design.compile(loss="binary_crossentropy",optimizer="adam",metrics=["accuracy"])

design.summary()

HISTORY_OF_MODEL=design.fit(train_X,train_Y,epochs=50,batch_size=64,validatio

n_data=(test_X,te

st_Y))

NP.save('/content/drive/MyDrive/stress_mini_project/Project/model/history.npy',HISTO

RY_OF_MODEL.

history)

design.save('/content/drive/MyDrive/stress_mini_project/Project/model/cnn_stress_mo

del1.h5')

from keras.models import load_model

load_design=load_model('/content/drive/MyDrive/stress_mini_project/Project/model/cn

n_stress_model1.h5')

cnn_accuracy=load_design.evaluate(test_X,test_Y)[1]

print("CNN Accuracy:",cnn_accuracy)

#predicting our cnn model with test data

PIC_TEST=random.randint(0,1999)

plt.imshow(test_X[PIC_TEST,:],cmap='gray')

plt.show()

prediction_s=load_design.predict(test_X[PIC_TEST,:].reshape(1,48,48,1))

print(prediction_s)
```

```
if prediction_s >0.5:

print("\n STRESS PERSON \n\n")

else:

print("\n NON STRESS PERSON \n\n")

y_pred = load_design.predict(test_X)>0.5

precision_CNN= precision_score(test_Y, y_pred, average='weighted')

recall_CNN = recall_score(test_Y, y_pred, average='weighted')

f1_CNN = f1_score(test_Y, y_pred, average='weighted')

# Print the results

print("Precision: {:.3f}".format(precision_CNN))

print("Recall: {:.3f}".format(recall_CNN)

print("F1 score: {:.3f}".format(f1_CNN))

import seaborn as sns

CM_OF_CNN= confusion_matrix(test_Y, y_pred, labels=[0, 1])

sns.set(font_scale=0.7)

sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')

plt.xlabel('Predicted labels')

plt.ylabel('True labels')

plt.title('CONFUSION MATRIX OF CONVOLUTION NEURAL NETWORK)

ti_label = ['NO STRESS, 'STRESS']

plt.xticks(NP.arange(len(ti_labels))+0.5, ti_label)

plt.yticks(NP.arange(len(ti_label))+0.5, tick_label)

plt.show()
```

```
from IPython.display import display, Javascript, Image

from google.colab.output import eval_js

from base64 import b64decode, b64encode

import cv2

import numpy as np

import PIL

import io

import html

import t

def js_to_img(js_reply):

IMG_bytes = b64decode(js_reply.split(',')[1])

JPG_as_np = np.frombuffer(IMG_bytes, dtype=NP.uint8)

PIC= cv2.imdecode(JPG_as_np, flags=1)

return PIC

base64 byte string

to be overlayed on video stream

def BBOX_to_bytes(bbox_array):

# CONVERT ARRY into PIL image

BBOX_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')

iobuf = io.BytesIO()

BBOX_PIL.save(iobuf, format='png')

# format return string

BBOX_BYTES = 'data:image/png;base64,{}'.format((str(b64encode(iobuf.getvalue())),
```

```
'utf-8')))

return BBOX_BYTES

F_cascade = cv2.CascadeClassifier(cv2.samples.findFile(cv2.data.haarcascades +

'haarcascade_frontalface_default.xml'))

def take_photo(filename='photo.jpg', quality=0.8):

JS_PIC= Javascript('''

async function takePhoto(quality) {

const DIV = document.createElement('DIV');

const capture = document.createElement('button');

capture.textContent = 'Capture';

DIV.appendChild(capture);

const video = document.createElement('video');

video.style.display = 'block';

const stream = await navigator.mediaDevices.getUserMedia({video: true});

document.body.appendChild(div);

div.appendChild(VIDEO);

VIDEO.srcObject = stream;

await VIDEO.play();

google.colab.output.setIframeHeight(document.documentElement.scrollHeight,

true);

await new Promise((resolve) => capture.onclick = resolve);

const CANVAS = document.createElement('CANVAS');

CANVAS.width = VIDEO.videoWidth;
```

```
CANVAS.height = VIDEO.videoHeight;
CANVAS.getContext('2d').drawImage(video, 0, 0);
stream.getVideoTracks()[0].stop();
div.remove();
return CANVAS.toDataURL('image/jpeg', quality);
}
''')
display(JS)
DATA = eval_js('takePhoto({})'.format(quality))
# get OpenCV format image
PIC = JS_to_img(DATA)
# grayscale img
GRAY_SCALE = cv2.cvtColor(PIC, cv2.COLOR_RGB2GRAY)
print(GRAY_SCALE.shape)
# get face bounding box coordinates using Haar Cascade
FACES = f_cascade.detectMultiScale(gray_scale)
# draw face bounding box on image
for (x,y,w,h) in FACES:
PIC= cv2.rectangle(PIC,(x,y),(x+w,y+h),(255,0,0),2)
cv2.imwrite(FILENAME, PIC)
return FILENAME
try:
filename = take_photo('photo.jpg')
```

```
print('Saved to {}'.format(FILENAME))

# Show the image which was just taken.

display(Image(FILENAME))

except Exception as ERR:

print(str(ERR))
24
import cv2

Frame=cv2.imread("/content/photo.jpg")

GRAY_s = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

NEW=GRAY_s/255

import matplotlib.pyplot as plt

#DATA STOREDin the `NEW` variable

plt.imshow(new, cmap='gray')

plt.show()

new_resized = cv2.resize(new, (48, 48), interpolation=cv2.INTER_LINEAR)

NEW= np.reshape(new_resized, (1, 48, 48, 1))

prediction_s = load_design.predict(new)

print(prediction_s)

if prediction_s > 0.5:

print("\n STRESS PERSON \n")

else:

print("\n NON STRESS PERSON \n")
18
import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

#DECISION TREE ACCEPT 2 D SHPAE

train_X=train_X.reshape(10000,48*48*1)

train_Y=train_Y.reshape(10000,1)

test_X=test_X.reshape(2000,48*48*1)

test_Y=test_Y.reshape(2000,1)

DTC = DecisionTreeClassifier()

DTC.fit(train_X, train_Y)

y_pred = DTC.predict(test_X)

DTC_ACC=accuracy_score(test_Y, y_pred)

print("Decision Tree Accuracy:",DTC_ACC)

PRECISION_DTC = precision_score(test_Y, y_pred, average='weighted')

RECALL_DTC = recall_score(test_Y, y_pred, average='weighted')

F1_D = f1_score(test_Y, y_pred, average='weighted')

print("Precision: {:.3f}".format(PRECISION_DTC))

print("Recall: {:.3f}".format(RECALL_DTC))

print("F1 score: {:.3f}".format(F1_D))

import seaborn as sns

CM_OF_DTC= confusion_matrix(test_Y, y_pred, labels=[0, 1])

sns.set(font_scale=0.7)

sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')

plt.xlabel('P')REDICTED LABELS

plt.ylabel('TRUE LABELS')
```

```
plt.title('CUNFUSION MATRIX OD DTC')

ti_label = ['No stress', 'stress']

plt.xticks(np.arange(len(ti_label))+0.5, ti_label)

plt.yticks(np.arange(len(ti_label))+0.5, ti_label)

plt.show()

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score

RFC = RandomForestClassifier(n_estimators=100, random_state=42)

RFC.fit(train_X, np.ravel(train_Y))

y_pred = RFC.predict(test_X)

RFC_ACC=accuracy_score(test_Y, y_pred)

print("Random Forest Accuracy:",RFC_ACC)

PRECISION_RFC = precision_score(test_Y, y_pred, average='weighted')

RECALL_RFC= recall_score(test_Y, y_pred, average='weighted')

F1_RFC= f1_score(test_Y, y_pred, average='weighted')

print("Precision: {:.3f}".format(PRECISION_RFC))

print("Recall: {:.3f}".format(RECALL_RFC))

print("F1 score: {:.3f}".format(F1_RFC))

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score

LR = LogisticRegression(max_iter=2000, random_state=42)

LR.fit(train_X, np.ravel(train_Y))

Vy_pred = LR.predict(test_X)
```

```
LOGISTIC_acc=accuracy_score(test_Y, y_pred)

print("Logistic Regression Accuracy:",LOGISTIC_acc)

precision_l = precision_score(test_Y, y_pred, average='weighted')

recall_l = recall_score(test_Y, y_pred, average='weighted')

f1_L= f1_score(test_Y, y_pred, average='weighted')

# Print the results

print("Precision: {:.3f}".format(precision_l))

print("Recall: {:.3f}".format(recall_l))

print("F1 score: {:.3f}".format(f1_l))

svm_s = SVC(kernel='rbf')

svm_s.fit(train_X, np.ravel(train_Y))

y_pred = svm_s.predict(test_X)

SVM_ACC=accuracy_score(test_Y, y_pred)

print("SVM Accuracy:",SVM_ACC)

# Calculate precision, recall, and F1 score

precision_svm = precision_score(test_Y, y_pred, average='weighted')

recall_svm = recall_score(test_Y, y_pred, average='weighted')

f1_svm = f1_score(test_Y, y_pred, average='weighted')

# Print the results

print("Precision: {:.3f}".format(precision_svm))

print("Recall: {:.3f}".format(recall_svm))

print("F1 score: {:.3f}".format(f1_svm))

CONTENT = [[SVM_acc,'SVM'],[LOG_acc,'LOGISTIC
```

```
REGRESSION'],[RFC_acc,"RANDOM FOREST "],[DTC_acc,"DECISSON
TREE"],[CNN_acc,"CNN"]]
CONTENT.sort()
LAB_CLASSIFICATIONS=list(zip(*data))[1]
ACCURACY_scores=list(zip(*data))[0]
ACCURACY_scores=[i*100 for i in ACCURACY_scores]
plt.figure(figsize=(9, 6))
plt.bar(labels, ACCURACY_scores, color=['red', 'green',
'blue','yellow','orange','purple','cyan'])
plt.title('ACCURACY SCORES OF DIFFERENT ALGORITHMS')
plt.xlabel('ALGORITHM TYPE')
plt.ylabel('ALG ACCURACY')
plt.ylim((0,100))
for i, v in enumerate(ACCURACY_scores):
 plt.text(i-0.25, v+0.02, str(round(v, 3))+"%", fontweight='bold',fontsize=12)
plt.show()
```

code_plgarism

ORIGINALITY REPORT

51%	44%	20%	49%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Northampton Student Paper	9%
2	Submitted to Sabanci Universitesi Student Paper	8%
3	Submitted to University of Strathclyde Student Paper	4%
4	Submitted to Universidad Internacional de la Rioja Student Paper	3%
5	Submitted to Instituto de Empress S.L. Student Paper	2%
6	Submitted to University of Southern California Student Paper	2%
7	Submitted to University of Rijeka - Faculty of Engineering Student Paper	2%
8	Submitted to The Robert Gordon University Student Paper	2%
9	hmkim312.github.io	

Internet Source

2%

10 www.stefanjaspers.com
Internet Source
2%

11 Submitted to The University of Memphis
Student Paper
2%

12 Submitted to King's College
Student Paper
2%

13 Submitted to Oberoi International School
Student Paper
1%

14 Submitted to National College of Ireland
Student Paper
1%

15 platzi.com
Internet Source
1%

16 Submitted to CSU, Chico
Student Paper
1%

17 Submitted to Queen Mary and Westfield College
Student Paper
1%

18 medium.com
Internet Source
1%

19 rc.library.uta.edu
Internet Source
1%

20 Submitted to City University
Student Paper

		1%
21	Submitted to University College London Student Paper	1%
22	developer.aliyun.com Internet Source	1%
23	kandi.openweaver.com Internet Source	1%
24	programming.vip Internet Source	1%
25	zir.nsk.hr Internet Source	1%
26	docplayer.net Internet Source	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		