ANALYSIS OF PSYCHOLOGICAL STRESS USING MACHINE LEARNING

Dr. NAVANEETHA KRISHNAN M, M.E. Ph.D., Head of the Department,

Department of Computer Science and Engineering

Mr.SACHIN RAJ G, B.E, Student of Computer Science Engineering

Mr.SANTHOSHKUMAR A, B.E, Student of Computer science and Engineering

St. Joseph College of Engineering, Sriperumbudur, Chennai.

Abstract

Stress plays an integral role in influencing one's decision-making capability, attention span, learning, and problem-solving capacity. Stress detection and modeling have been active areas of research in the fields of psychology and computer science in recent times. Psychologists quantify stress using affective states, which is the experience of feeling the underlying emotional state. Most of the work in classifying human stress was achieved using user-dependent models, incapable of generalizing to a new user.

This causes new user to spend a significant amount of their time in training the model to predict their affective states. There is an urgent need to treat basic mental health problems that prevail among children which may lead to complicated problems, if not treated at an early stage. Machine learning Techniques are currently well suited for analyzing medical data and diagnosing the problem.

The attributes have been reduced by analysing Features over the full attribute data set. The accuracy over the selected attribute set on various machine learning algorithms have been compared.

Key Terms: BMI - Stress Prediction ,Stages ,Machine Learning ,Support Vector Machine algorithm ,Adaboost ,MLP classifier ,Decision tree algorithm ,Psychological Analysis.

Introduction

The stress-detection system is proposed based on physiological signals. Parameters like galvanic skin response (GSR), stress rate (HR), Body temperature, Muscle tension, Blood pressure are proposed to provide information on the state of mind of an individual, due to their non-intrusiveness and non invasiveness.

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labelling to learn data has to be labelled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Literature Survey

The paper "Joint-Channel-Connectivity-Based Feature Selection and Classification on fNIRS for Stress Detection in Decision-Making" By Meiyan Huang, Xiaoling Zhang, Xiumei Chen, Yiling Mai, Xiaohua Wu, Jiubo Zhao, and Qianjin Feng in 2022, This approach integrates feature selection and classifier modeling into a sparse model, where intra- and inter-channel regularizers are designed to explore potential correlations among channels to obtain discriminating features.

The paper “Human stress classification during public speaking using physiological signals” By Aamir Arsalan, Muhammad Majid in 2021, The selected set of features from all modalities are fused to classify the stress into two classes. Classification is performed via a leave-one-out cross-validation scheme by using five different classifiers. The highest accuracy of 96.25% is achieved using a support vector machine classifier with radial basis function. Statistical analysis is performed to examine the significance of EEG

The paper “Unlocking stress and forecasting its consequences with digital technology” By Sarah M. Goodday and Stephen Friend in 2019 The growth and availability of digital technologies involving wearable devices and mobile phone apps afford the opportunity to dramatically improve measurement of the biological stress response in real time. In parallel, the advancement and capabilities of artificial intelligence (AI) and machine learning could discern heterogeneous, multidimensional information from individual signs of stress, and possibly inform how these signs forecast the downstream consequences of stress in the formofend - organdamage

.

System Design

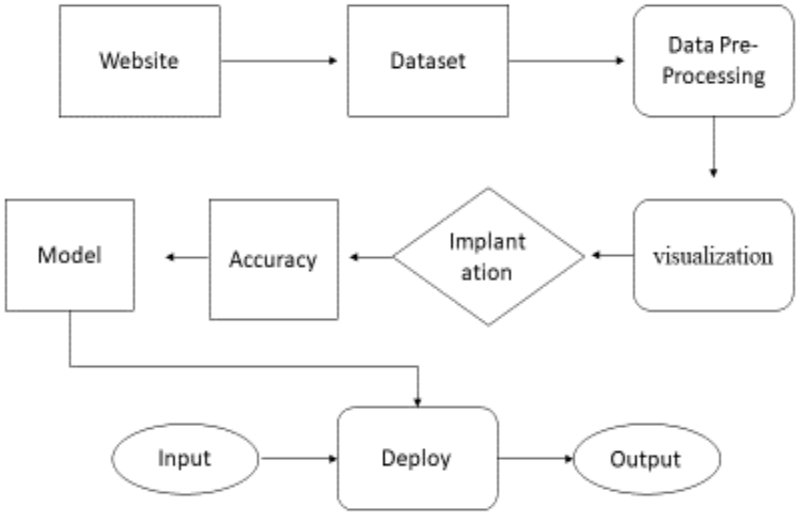
The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using Random Forest, logistic, Decision tree algorithms and Support vector classifier (SVC) are applied on the Training set and based on the test result accuracy, Test set prediction is done.

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be done.

The prediction of Human Stress, a high accuracy prediction model is effective because of the following reasons: It provides better results in classification problem.

It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables. It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.

SYSTEM ARCHITECTURE



Machine learning needs data gathering have lot of past data's. Data gathering have sufficient historical data and raw data. Before data pre-processing, raw data can't be used directly. It's used to pre-process then, what kind of algorithm with model. Training and testing this model working and predicting correctly with minimum errors. Tuned model involved by tuned time to time with improving the accuracy.

.

CODING

Module – 1

Pre-Processing

In [ ]:

```
import pandas as p
import numpy as n
```

In [ ]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
data=p.read_csv('human.csv')
```

In [ ]:

```
data.head()
```

In [ ]:

```
data.shape
```

In [ ]:

```
data.size
```

In [ ]:

```
data.isnull()
```

In [ ]:

```
data.columns
```

In [ ]:

```
data['sl'].unique()
```

In [ ]:

```
data[data["sl"] == 2]
```

In [ ]:

```
data[data["sl"] == 3]
```

In [ ]:

```
a = data["bo"].unique()
a.sort()
print(a)
```

In [ ]:

```
data["hr"].unique()
```

In [ ]:

```
data["sr"].unique()
```

In [ ]:

```
df = data.dropna()
```

In [ ]:

```
df
```

In [ ]:

```
df.isnull().sum()
```

In [ ]:

```
df.shape
```

In [ ]:

```
print("Maximum Age :",df["rem"].max())
print("Minimum Age :",df["rem"].min())
print("Mean of Age :","%.2f" % df["rem"].mean())
print("Median of Age :","%.2f" % df["rem"].median())
```

In [ ]:

```
df.describe()
```

In [ ]:

```
df.corr()
```

In [ ]:

```
df.info()
```

In [ ]:

```
p.crosstab(df["rem"], df["sl"])
```

In [ ]:

```
df.groupby(["sr.1","sl"]).groups
```

In [ ]:

```
df["sl"].value_counts()
```

In [ ]:

```
df["hr"].value_counts()
```
In [ ]:

```
p.Categorical(df["rr"]).describe()
```
In [ ]:

```
p.Categorical(df["sr.1"]).describe()
```
In [ ]:

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

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

Module - 2

Visualization

In [ ]:

```
import pandas as p
import numpy as n
import seaborn as s
import matplotlib.pyplot as plt
```
In [ ]:

```
import warnings
warnings.filterwarnings('ignore')
```
In [ ]:

```
data=p.read_csv('human.csv')
```
In [ ]:

```
data
```
In [ ]:

```
df=data.dropna()
```
In [ ]:

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

```
p.crosstab(df.sl,df.rem)
```
In [ ]:

```
#Histogram
df['rem'].hist(figsize=(10,5), color='m', alpha=1)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('EYE MOVEMENT')
```

In [ ]:

```
plt.figure(figsize=(16,4))
plt.subplot(1,2,1)
plt.hist(df['rr'])
plt.title("RESPIRATION RATE")
plt.xlabel("X")
plt.ylabel("Y")
plt.subplot(1,2,2)
plt.hist(df['t'])
plt.xlabel('X')
plt.title('BODY TEMPERATURE')
plt.ylabel("Y")
plt.show()
```

In [ ]:

```
plt.figure(figsize=(9,6))
s.scatterplot(x=df['t'],y=df["rr"])
plt.show()
```

In [ ]:

```
plt.figure(figsize=(12,8))
plt.scatter(df["rem"],df["hr"],color="red")
```

In [ ]:

```
import seaborn as s
s.boxplot(df['sr'], color='c')
```

In [ ]:

```
import seaborn as s
s.boxplot(df['lm'], color='y')
```

In [ ]:

```
#Density Subplot
plt.figure(figsize= (18,12))
plt.subplot(3,3,1)
df["sl"].plot(kind='density')
plt.title("sl")
plt.subplot(3,3,2)
df["bo"].plot(kind='density')
plt.title("bo")
plt.subplot(3,3,3)
df["lm"].plot(kind='density')
```

```
plt.title("lm")
plt.subplot(3,3,4)
df["t"].plot(kind='density')
plt.title("t")
plt.subplot(3,3,5)
df["rr"].plot(kind='density')
plt.title("rr")
plt.subplot(3,3,6)
df["rem"].plot(kind='density')
plt.title("rem")
plt.subplot(3,3,7)
df["sr.1"].plot(kind='density')
plt.title("sr.1")
plt.subplot(3,3,8)
df["hr"].plot(kind='density')
plt.title("hr")
plt.subplot(3,3,9)
df["sr"].plot(kind='density')
plt.title("sr")

plt.show()
```

In [ ]:

```
df.hist(figsize=(15,55),layout=(15,4))
plt.show()
```

In [ ]:

```
fig, ax = plt.subplots(figsize=(10,5))
s.stripplot(y = df['sl'], x = df['lm'], ax=ax)
plt.title("HUMAN STRESS CLASSFICATION")
plt.show()
```

In [ ]:

```
s.pairplot(df)
plt.show()
```

In [ ]:

```
def sl(df, variable):
    dataframe_pie = df[variable].value_counts()
    ax = dataframe_pie.plot.pie(figsize=(9,9), autopct='%1.2f%%', fontsize = 10)
    ax.set_title(variable + ' \n', fontsize = 10)
    return n.round(dataframe_pie/df.shape[0]*100,2)

sl(df, 'sl')
```

In [ ]:

```
plt.plot(df["sl"], df["sr"], color='m')
```

```
plt.xlabel('sl')
plt.ylabel('sr')
plt.title('HUMAN STRESS CLASSFICATION')
plt.show()
```

In [ ]:

In [ ]:

In [ ]:

Module - 3
SUPPORT VECTOR MACHINE

In [ ]:

```
import pandas as p
import numpy as n
import matplotlib.pyplot as plt
import seaborn as s
```

In [ ]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
data=p.read_csv('human.csv')
data.head()
```

In [ ]:

```
data.columns
```

In [ ]:

```
df=data.dropna()
```

In [ ]:

```
df
```

In [ ]:

```
df.columns
```

In [ ]:

```
df.tail()
```

In [ ]:

```
df.shape
```

In [ ]:

```
df.info()
```

In [ ]:

```
df
```

In [ ]:

```
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='sl', axis=1)
#Response variable
y = df.loc[:,'sl']
```

In [ ]:

```
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros =RandomOverSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(X,y)
print("OUR DATASET COUNT         : ", Counter(y))
print("OVER SAMPLING DATA COUNT  : ", Counter(y_ros))
```

In [ ]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_ros, y_ros, test_size=0.20, random_state=1,
stratify=y_ros)
print("Number of training dataset : ", len(X_train))
print("Number of test dataset     : ", len(X_test))
print("Total number of dataset    : ", len(X_train)+len(X_test))
```

In [ ]:

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

In [ ]:

```
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score

s = SVC()

s.fit(X_train,y_train)

predictS = s.predict(X_test)

print("")
print('Classification report of Support Vector Machine Result is:')
print("")
print(classification_report(y_test,predictS))
print("")
```

```
cm=confusion_matrix(y_test,predictS)
print('Confusion Matrix result of Support Vector Machine is:\n',cm)
print("")

sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity )
print("")
specificity = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity)
print("")

accuracy = cross_val_score(s, x_ros, y_ros, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)


print("")
print("Accuracy Result of Support Vector Machine is:",accuracy.mean() * 100)
svc=accuracy.mean() * 100
```

In [ ]:

```
TP = cm[0][0]
FP = cm[1][0]
FN = cm[0][1]
TN = cm[1][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)
```

In [ ]:

```
def plot_confusion_matrix(cm, title='Confusion matrix-Support Vector Machine',
cmap=plt.cm.cool):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

cm1=confusion_matrix(y_test, predictS)
print('Confusion matrix-Support Vector Machine:')
print(cm)
plot_confusion_matrix(cm)
```

In [ ]:

Module - 4
ADABOOST CLASSIFIER

In [ ]:

```
import pandas as p
import numpy as n
import matplotlib.pyplot as plt
import seaborn as s
```

In [ ]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
data=p.read_csv('human.csv')
```

In [ ]:

```
data.columns
```

In [ ]:

```
df=data.dropna()
```

In [ ]:

```
dfss
```

In [ ]:

```
df.columns
```

In [ ]:

```
df.head()
```

In [ ]:

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

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

```
df
```
In [ ]:

```
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='sl', axis=1)
#Response variable
y = df.loc[:,'sl']
```
In [ ]:

```
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros =RandomOverSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(X,y)
print("OUR DATASET COUNT        : ", Counter(y))
print("OVER SAMPLING DATA COUNT  : ", Counter(y_ros))
```
In [ ]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_ros, y_ros, test_size=0.20, random_state=1,
stratify=y_ros)
print("Number of training dataset : ", len(X_train))
print("Number of test dataset     : ", len(X_test))
print("Total number of dataset    : ", len(X_train)+len(X_test))
```
In [ ]:

```
#According to the cross-validated MCC scores, the random forest is the best-performing model,
so now let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```
In [ ]:

```
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import cross_val_score

ADC = AdaBoostClassifier()

ADC.fit(X_train,y_train)

predictRF = ADC.predict(X_test)
```

```
print("")
print('Classification report of AdaBoostClassifier Result is:')
print("")
print(classification_report(y_test,predictRF))
print("")

cm=confusion_matrix(y_test,predictRF)
print('Confusion Matrix result of AdaBoostClassifier is:\n',cm)
print("")

sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity )
print("")
specificity = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity)
print("")

accuracy = cross_val_score(ADC, x_ros, y_ros, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)


print("")
print("Accuracy Result of AdaBoostClassifier is:",accuracy.mean() * 100)
adc=accuracy.mean() * 100
```

In [ ]:

```
def Bar_Chart():
    import matplotlib.pyplot as plt
    data=[adc]
    alg="AdaBoostClassifier"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("Red"))
    plt.title("Accuracy Result of AdaBoostClassifier",fontsize=15)
    plt.legend(b,data,fontsize=9)
Bar_Chart()
```

In [ ]:

```
TP = cm[0][0]
FP = cm[1][0]
FN = cm[0][1]
TN = cm[1][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
```

```
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)
```

In [ ]:

```
def plot_confusion_matrix(cm, title='Confusion matrix-AdaBoostClassifier',
cmap=plt.cm.Accent):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

cm1=confusion_matrix(y_test, predictRF)
print('Confusion matrix AdaBoostClassifier:')
print(cm)
plot_confusion_matrix(cm)
```

In [ ]:

Module - 5
MLP CLASSIFIER

In [ ]:

```
import pandas as p
import numpy as n
import matplotlib.pyplot as plt
import seaborn as s
```

In [ ]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
data=p.read_csv('human.csv')
```
In [ ]:

```
data.columns
```
In [ ]:

```
df=data.dropna()
```
In [ ]:

```
df
```
In [ ]:

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

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

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

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

```
df
```
In [ ]:

```
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='sl', axis=1)
#Response variable
y = df.loc[:,'sl']
```
In [ ]:

```
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros =RandomOverSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(X,y)
print("OUR DATASET COUNT        : ", Counter(y))
print("OVER SAMPLING DATA COUNT  : ", Counter(y_ros))
```
In [ ]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_ros, y_ros, test_size=0.20, random_state=1,
stratify=y_ros)
print("Number of training dataset : ", len(X_train))
print("Number of test dataset     : ", len(X_test))
```

```
print("Total number of dataset    : ", len(X_train)+len(X_test))
```

In [ ]:

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

In [ ]:

```
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import cross_val_score

mlp = MLPClassifier()

mlp.fit(X_train,y_train)

predictLR = mlp.predict(X_test)

print("")
print('Classification report of MLPClassifier Result is:')
print("")
print(classification_report(y_test,predictLR))
print("")

cm=confusion_matrix(y_test,predictLR)
print('Confusion Matrix result of MLPClassifier is:\n',cm)
print("")

sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity )
print("")
specificity = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity)
print("")

accuracy = cross_val_score(mlp, x_ros, y_ros, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)

print("")
print("Accuracy Result of MLPClassifier is:",accuracy.mean() * 100)
mlp=accuracy.mean() * 100
```

In [ ]:

```
def Bar_Chart():
    import matplotlib.pyplot as plt
    data=[mlp]
    alg="MLPClassifier"
    plt.figure(figsize=(5,5))
```

```
    b=plt.bar(alg,data,color=("m"))
    plt.title("Accuracy Result of MLPClassifier",fontsize=15)
    plt.legend(b,data,fontsize=9)
Bar_Chart()
```

In [ ]:

```
TP = cm[0][0]
FP = cm[1][0]
FN = cm[0][1]
TN = cm[1][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)
```

In [ ]:

```
def plot_confusion_matrix(cm, title='Confusion matrix-MLPClassifier', cmap=plt.cm.autumn):
    plt.imshow(cm, interpolation='hermite', cmap=cmap)
    plt.title(title)
    plt.colorbar()

cm1=confusion_matrix(y_test, predictLR)
print('Confusion matrix-MLPClassifier:')
print(cm)
plot_confusion_matrix(cm)
```

In [ ]:

Module - 6

DECISION TREE CLASSIFIER

In [ ]:

```
import pandas as p
import numpy as n
import matplotlib.pyplot as plt
import seaborn as s
```

In [ ]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
data=p.read_csv('human.csv')
```

In [ ]:

```
data.columns
```

In [ ]:

```
df=data.dropna()
df = df.rename({'sr.1':'srh'},axis=1)
```

In [ ]:

```
df
```

In [ ]:

```
df.columns
```

In [ ]:

```
df.head()
```

In [ ]:

```
df.shape
```

In [ ]:

```
df.info()
```

In [ ]:

```
df
```

In [ ]:

```
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='sl', axis=1)
#Response variable
y = df.loc[:,'sl']
```

In [ ]:

```
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros =RandomOverSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(X,y)
print("OUR DATASET COUNT        : ", Counter(y))
print("OVER SAMPLING DATA COUNT  : ", Counter(y_ros))
```

In [ ]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_ros, y_ros, test_size=0.20, random_state=1,
stratify=y_ros)
print("Number of training dataset : ", len(X_train))
print("Number of test dataset     : ", len(X_test))
print("Total number of dataset    : ", len(X_train)+len(X_test))
```

In [ ]:

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

In [ ]:

```
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score

dt = DecisionTreeClassifier()

dt.fit(X_train,y_train)

predictDT = dt.predict(X_test)

print("")
print('Classification report of Decision Tree Result is:')
print("")
print(classification_report(y_test,predictDT))
print("")

cm=confusion_matrix(y_test,predictDT)
print('Confusion Matrix result of Decision Tree is:\n',cm)
print("")

sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity )
print("")
specificity = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity)
print("")

accuracy = cross_val_score(dt, x_ros, y_ros, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)


print("")
print("Accuracy Result of Decision Tree is:",accuracy.mean() * 100)
DTC=accuracy.mean() * 100
```

In [ ]:

```
def Bar_Chart():
```

```
    import matplotlib.pyplot as plt
    data=[DTC]
    alg="Decision Tree"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("Violet"))
    plt.title("Accuracy Result of Decision Tree",fontsize=15)
    plt.legend(b,data,fontsize=9)
Bar_Chart()
```

In [ ]:

```
TP = cm[0][0]
FP = cm[1][0]
FN = cm[0][1]
TN = cm[1][1]
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)
```

In [ ]:

```
def plot_confusion_matrix(cm, title='Confusion matrix-Decision Tree', cmap=plt.cm.BuPu):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

cm1=confusion_matrix(y_test, predictDT)
print('Confusion matrix-Decision Tree:')
print(cm)
plot_confusion_matrix(cm)
```

In [ ]:

```
import joblib
joblib.dump(dt,'model.pkl')
```

In [ ]:

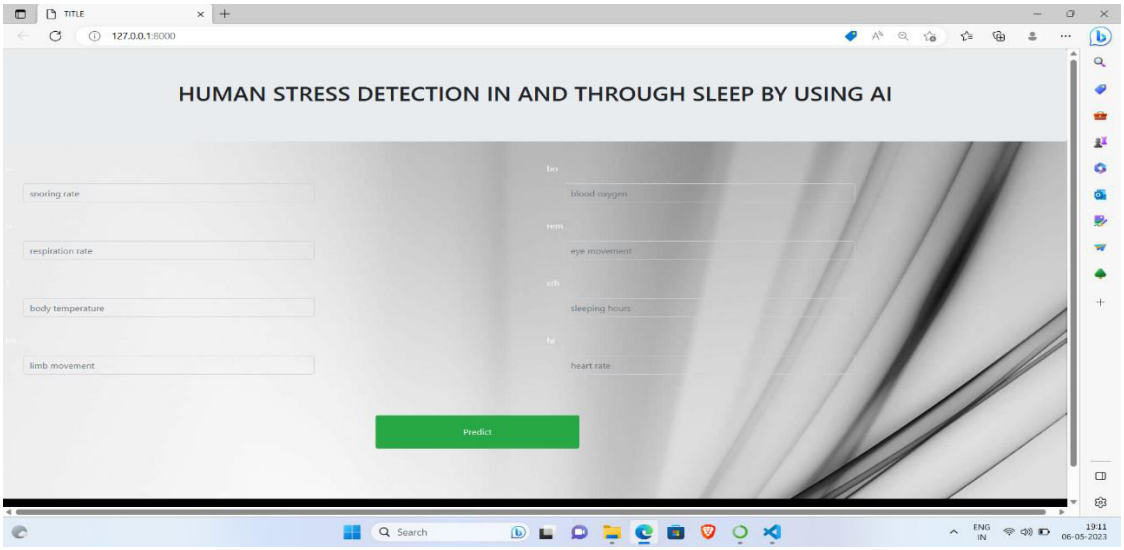
In [ ]:

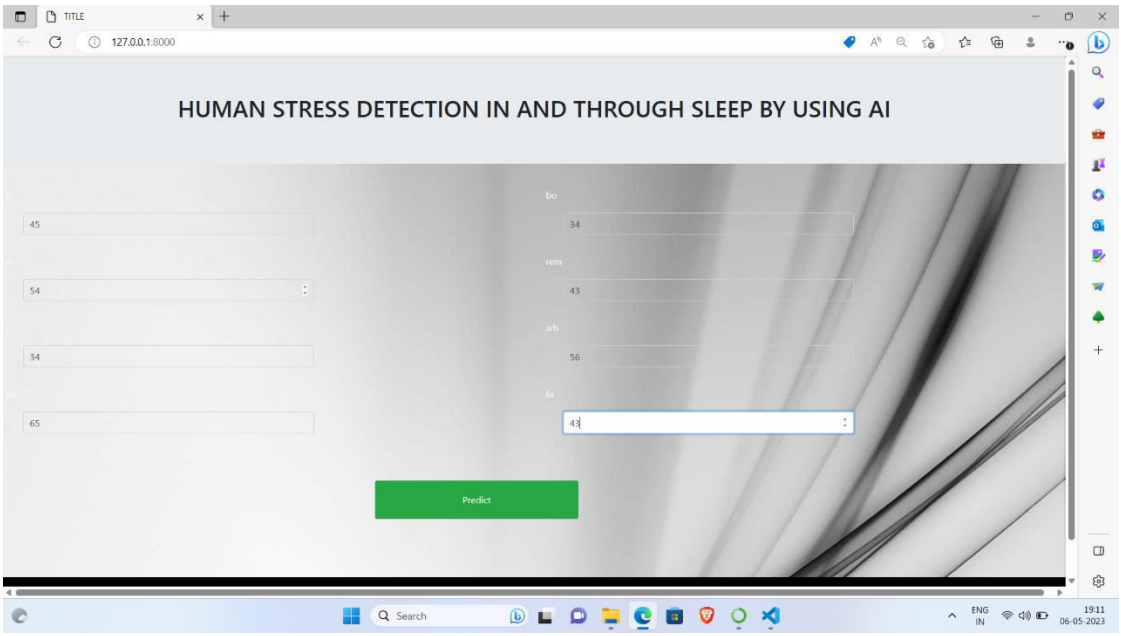
Deploy

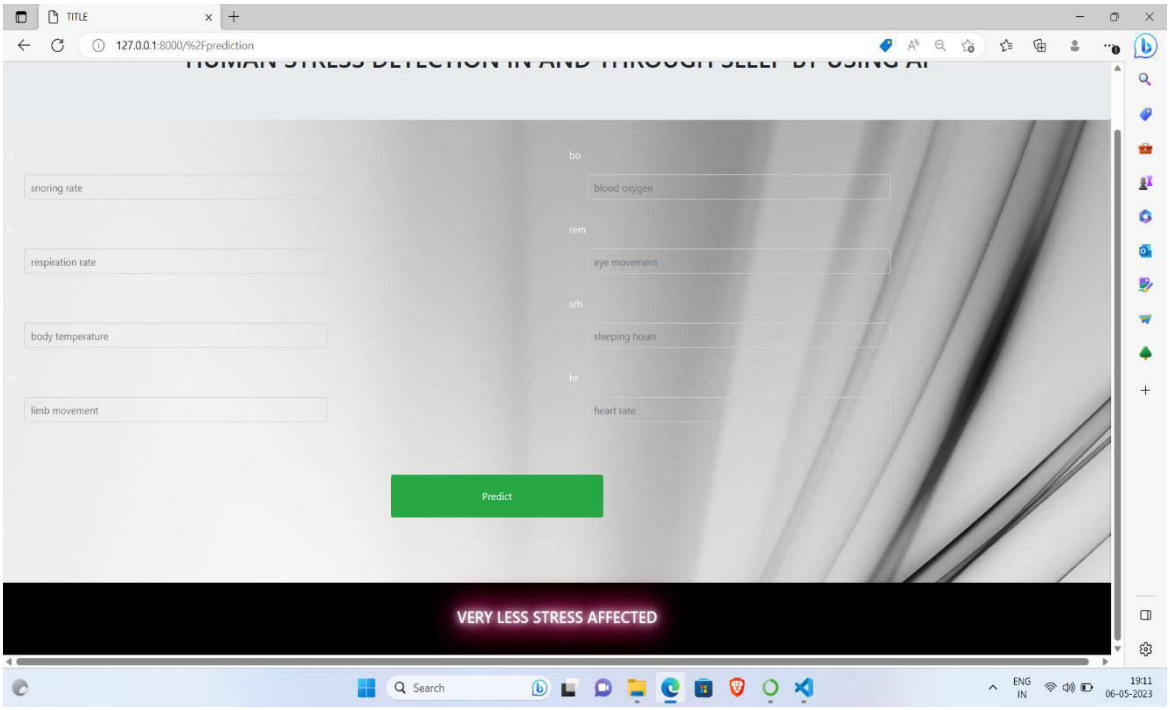
```
from django.shortcuts import render

from django.shortcuts import render, redirect

import numpy as np

import joblib


model =
joblib.load('C:/Users/SPIRO15/Desktop/boo/Project/ML/ITPML27/deploy/app/model.pkl')


# Create your views here.

def home(request):

    return render(request, "index.html")


def predict(request):

    if request.method == "POST":

        int_features = [x for x in request.POST.values()]

        int_features = int_features[1:]

        print(int_features)

        final_features = [np.array(int_features, dtype=object)]

        print(final_features)

        prediction = model.predict(final_features)

        print(prediction)
```

```
    output = prediction[0]
    print(f'output{output}')
    if output == 0:
        return render(request, 'index.html', {"prediction_text":"VERY LESS STRESS
AFFECTED"})
    elif output == 1:
        return render(request, 'index.html', {"prediction_text":"LESS STRESS AFFECTED"})
    elif output == 2:
        return render(request, 'index.html', {"prediction_text":"NORMAL OR MODERATE
STRESS AFFECTED"})
    elif output == 3:
        return render(request, 'index.html', {"prediction_text":"MILD STRESS AFFECTED"})
    else:
        return render(request, 'index.html', {"prediction_text":"VERY HIGH STRESS
AFFECTED"})
    print(output)
```

SNAPSHOTS


TITLE
127.0.0.1:8000
HUMAN STRESS DETECTION IN AND THROUGH SLEEP BY USING AI
snoring rate
blood oxygen
respiration rate
eye movement
body temperature
sleeping hours
limb movement
heart rate
Predict
Search
ENG
IN
19:11
06-05-2023


TITLE
127.0.0.1:8000
HUMAN STRESS DETECTION IN AND THROUGH SLEEP BY USING AI
45
34
54
43
34
56
65
43
Predict
Search
ENG
IN
19:11
06-05-2023



CONCLUSION

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set of higher accuracy score algorithm will be find out. The founded one is used in the application which can help to find the Human Stress of the patient.

.

FUTURE ENHANCEMENTS

- ❖ Deploying the project in the cloud.
- ❖ To optimize the work to implement in the IOT system.

REFERENCES:

[1] Meiyan Huang, Xiaoling Zhang, Xiumei Chen, Yiling Mai, Xiaohua Wu, Jiubo Zhao, and Qianjin Feng, “Joint-Channel-Connectivity-Based Feature Selection and Classification on fNIRS for Stress Detection in Decision-Making” IEEE(2022)

[2] Aamir Arsalan, Muhammad Majid, " Human stress classification during public speaking using physiological signals”

[3] Sarah M.Goodday and Stephen Friend, “Unlocking stress and forecasting its consequences with digital technology”

[4] Hatoon AlSagri, Mourad Ykhlef, “Machine Learning-based Approach for Depression Detection in Twitter Using Content and Activity Features.”

[5] Meera Sharma, Sonok Mahapatra, “Predicting the Utilization of Mental Health Treatment with Various Machine Learning Algorithms”

[6] Ms. Sumathi M.R, Dr. B. Poorna , “Prediction of Mental Health Problems Among Children Using Machine Learning Techniques”

[7] Sandhya P, Mahek Kantesaria, “Prediction of Mental Disorder for employees in IT Industry”

AUTHOR 1

Dr.M.Navaneethakrishnan M.E., PhD is a Head of the Department in the Department of Computer Science and Engineering at St. Joseph College of Engineering, Sriperumbudur, Chennai, Tamil Nadu. He has completed his Ph.D, in Cyber Security - Computer Science and Engineering in 2017 from Manonmaniam Sundaranar University (MSU) Tirunelveli, Tamilnadu. He has done his M.E, CSE in Anna University Chennai in the year 2008. Dr.M.Navaneethakrishnan has 15 years of teaching experience and has 58 publications in International Journals and Conferences. His research interests include network security, Computer Networks, data science and Machine Learning. He is an active member of ISTE, CSI, IEANG and IEI

AUTHOR 2

Mr.Sachin Raj G B.E., Student Of Computer Science and Engineering At St.Joseph College Of Engineering, Sriperumbudur, Chennai, Tamilnadu. I had Attended Many Workshops, Seminars in Python, Machine Learning. I Completed Python Full Stack Course In Besant Technologies.

AUTHOR 3

Mr. Santhoshkumar A B.E., Student Of Computer Science and Engineering At St.Joseph College Of Engineering, Sriperumbudur, Chennai, Tamilnadu. I had Attended Many Workshops and Seminars in The Area Of Python and Machine Learning. I Got Placed in Qspider Technologies