

**A REPORT**  
**ON**  
**STRESS DETECTION IN AUTOMOBILE DRIVERS USING**  
**PHYSIOLOGICAL FEATURES**

**BY**

**Name of the:** Vivek Bindal  
**Student**

**ID.No. :**  
2018HT12641

**Dissertation work carried out at**  
**Ebizon, Noida**



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**  
**(March, 2020)**

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**

**(March, 2020)**

**A  
REPORT**

**ON**

**STRESS DETECTION IN AUTOMOBILE DRIVERS USING  
PHYSIOLOGICAL FEATURES**

**BY**

**Name of: Vivek Bindal  
the Student**

**ID.No. :  
2018HT12641**

**Discipline :  
Software Systems**

Prepared in partial fulfilment of the  
WILP Dissertation/Project/Project Work Course

**AT**

**Ebizon, Noida**



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**

**(March, 2020)**

## **ACKNOWLEDGEMENT**

First and foremost, we would like to express our deepest gratitude to our supervisor, Mr. Utkarsh Srivastava, for giving us the golden opportunity to do this project and guiding us throughout the project. His motivation has allowed us to complete the project in limited time.

Finally we would like to thank our family and friends for their support during the completion of the project.

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE  
PILANI (RAJASTHAN)  
WILP Division**

**Organization: ...Ebizon.....Location: ...Noida.....**

**Duration .....3 months..... Date of Start.....17-Jan-2020.....**

**Date of Submission .....31-March-2020.....**

**Title of the Project:** STRESS DETECTION IN AUTOMOBILE DRIVERS USING  
PHYSIOLOGICAL FEATURES

**ID No./Name of the student:** Vivek Bindal(2018HT12641)

**Name (s) and Designation (s) of**

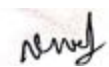
**your Supervisor and Additional Examiner:** Mr. Utkarsh Srivastava and Mr. Ashish Singh

**Name of the Faculty Mentor:** Mr. Venkat

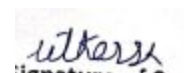
**Key Words:** Data Analysis, Artificial Intelligence

**Project Areas:** Artificial intelligence, Information retrieval and extractions techniques.

**Abstract:** This project puts forward methods analyzing physiological data during real-world automobile driving tasks to determine the cognitive stress drivers. In the dataset used, galvanic skin response signals of 15 drivers were recorded continuously who followed a set route that experienced open routes as well as heavy traffic conditions. The duration of the drive was of at least 50 minutes; but this, however, varied depending upon the traffic situation. The analysis was done over a span of 5 minutes for all three levels of stress: high, medium and low, in different areas of the route which included the city, highway and the driver being completely at rest. These sets of data were then used to extract various features, like mean, variance, frequency of spikes exceeding a particular threshold, area under spikes, sum of durations of rise to local maxima and sum of rise in magnitudes during spikes. These were evaluated using neural networks and Support Vector Machine (SVM) and the results were compared. We found out that SVM gives better results in this application than neural networks.



Signature Of the Student



Signature of Supervisor

## List of Figures

S. No.	Figure Caption	Page no.
1.	Overview of Proposed Methodology	11

## List of Tables

S. No.	Table Heading	Page No.
1.	Feature matrix of Foot GSR	14
2.	Feature matrix of Hand GSR	15
3.	Accuracy from different classifiers	15

## List of Abbreviations

S. No.	Abbreviated Form	Full Form
1.	ANN	Artificial Neural Network
2.	ANOVA	Analysis of Variance
3.	ECG	Electrocardiogram
4.	EDA	Electrodermal Activity
5.	EDR	ECG derived respiration
6.	EMG	Electromyogram
7.	GSR	Galvanic Skin Response
8.	HRV	Heart Rate Variability
9.	IMU	Inertial Motion Unit
10.	LDA	Linear Discriminant Analysis

11.	LDA	Nearest Mean Classifier
12.	PPG	Photoplethysmogram
13.	PPG	Power Spectral Density
14.	SVM	Support Vector Machine

## Contents

1.	INTRODUCTION.....	8
2.	LITERATURE REVIEW.....	9
3.	METHODOLOGY.....	10
	i)     Algorithm A: Algorithm to create feature matrix.....	13
4.	RESULT AND DISCUSSION.....	14
	i)     Feature matrix of foot GSR.....	14
	ii)    Feature matrix of hand GSR.....	15
	iii)   Accuracy from different classifiers.....	15
5.	SOURCE CODE.....	16
	i)     Segmentation function(segmentation.py).....	16
	ii)    Function to calculate local maxima and minima (maxmin.py).....	21
	iii)   Function include peaks exceeding particular threshold(peaks_func.py).	23
	iv)    Function to normalise signal (normalise.py).....	25
	v)     Function to extract features (features_func.py).....	26
	vi)    Main function (main.py).....	28
	vii)   Machine learning algorithms.....	29
6.	CONCLUSION.....	29
7.	PLAN OF WORK.....	30
8.	REFERENCES.....	30

## 1. INTRODUCTION

Stress recognition in automobile drivers is a growing concern because of an increase in the number of onboard electronic appliances, like cell-phones, audio and video players, navigation systems, which can lead to distraction of the driver's attention from driving. In driving circumstances, when there is high traffic and the driver experiences a higher degree of stress, automated management of the appliances may be desired. Understanding frustration of drivers has been listed as one of the vital areas for the improvement of intelligent transportation systems.

Various protocols can be used to measure driver workload. These include eye glands and on road matrix, but these parameters are very costly and difficult to maintain. Therefore, as an alternative, we can use physiological sensors to obtain electrodermal activity that can be processed by an automated on board system. This system can give us an indication of the driver's internal state under natural driving conditions without causing any obstruction while driving.

Stress is mainly of two types, "eustress" and "distress" [1], where eustress is positive stress such as bliss. However, for the purpose of our project, we are referring to stress only as distress, with a negative bias, that causes an increase in driver workload. A number of studies show that highly stressed situations lead to diminishing decision capabilities and lower situational awareness in addition with degraded performance that could damage driving ability.

Electrodermal Activity (EDA) is a measure of variation in conductivity of the skin. The variation occurs with the changing state of sweat glands in our skin, when we are physiologically aroused, sweat glands get activated, hence, sweat is released and the conductivity increases. EDA is also known as skin conductance, galvanic skin response(GSR) and electrodermal response. We can say that galvanic skin response is a



measure of emotional and sympathetic responses. This is measured by recording electric resistance between two electrodes, when a weak current is passed between the two.

In this project we are making an attempt to identify the physiological signals which can be used as a powerful metric for stress detection. These signals can be obtained continuously without interference in driving. This information can be supplied to the automated systems that can help the driver to cope up with stress. This may include management of non critical on board electronic systems. Calls and messages can be set to silent mode, navigation systems can be set to provide only vital information during high stress situations. The volume of the music player can be lowered or relaxing tunes can be played to ease the driver. On the contrary the driver can be presented with more entertainment options during low stress.

In the Physionet dataset used [2], the signals are gathered for a drive, which includes regions of low, medium and high stress. The driver rests in the garage for the initial 15 minutes and then drives through the city traffic. He, then, reaches the highway, crosses some tolls, makes a U-turn and traces his way back to the garage. For the purpose of feature extraction, 5 minutes of data is extracted from each of the three driving situations, that is, city, highway and rest. These are a set of non overlapping 5 minutes that is done to decrease the amount of computation needed to detect stress.

## **2. LITERATURE REVIEW**

In [3], Nearest Mean Classifier (NMC) was developed to study the dynamic change in driving style in accordance with the stress condition of the driver. This is done by monitoring mechanical and physiological features. The mechanical features used are steering wheel angle and latent force; and electrocardiogram (ECG) and ECG derived respiration (EDR) were used to calculate physiological features. They achieved an overall accuracy of 90% with their algorithm. In [4], the authors tried to minimize the hardware required for data collection by using only electromyogram (EMG) signals to assess

mental stress. After extracting features, analysis of variance (ANOVA), developed by Fischer Projection, is used for classification. With this algorithm, an overall accuracy of 92% was achieved.

Some people made an attempt to predict the driver's stress level by evaluating the movement pattern of the steering wheel only, as in [5]. Physiological features were not used to assess the stress level. Movement pattern was assessed by using an inertial motion unit sensor, which was placed on a glove worn by the driver. Steering wheel movement was used as a parameter for stress detection and Support Vector Machine (SVM) was used as a classifier and an efficiency of 94.78% was achieved. An attempt to combine physical movements and physiological features was again made in [6], where the researchers used photoplethysmogram (PPG) signal, measured from fingertip, and steering wheel angle data, derived from Inertial Motion Unit (IMU) sensors' reading. By using SVM as a classifier, efficiency of 95% was obtained.

In [7], the authors used PPG, heart rate and GSR signals for features extraction and then self-organising maps are used to classify them into low, median and high stress. With the proposed approach they achieved an efficiency of 81.6%. Using a lesser number of features, an overall efficiency of 97.4% was achieved in [8]. In this, the researchers used GSR and heart rate variability (HRV) for feature extraction. These were analysed using Fischer projection matrix and Linear Discriminant Analysis (LDA).

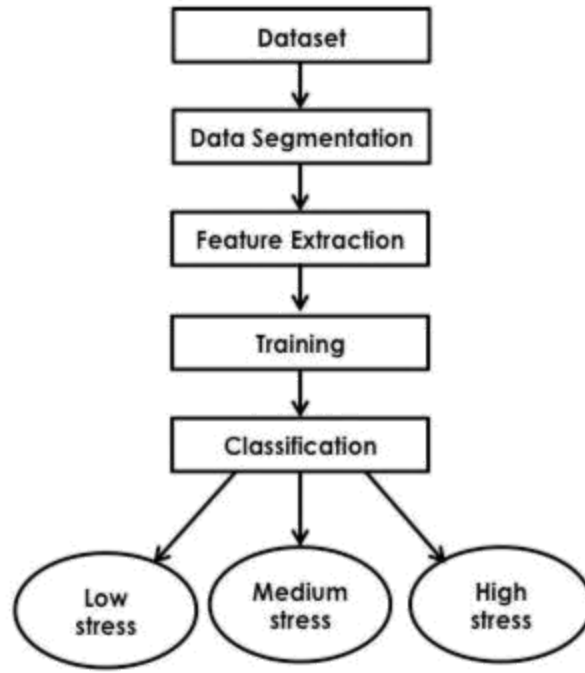
### **3. METHODOLOGY**

The algorithm uses 5 minutes intervals of non overlapping sets from rest, city and highway representing low, high and medium stress levels respectively. To ensure the correctness of the algorithm it is necessary to choose these sets carefully. The last 5 minutes from the 15 minutes of rest conditions were used. The span on the highway consists of the time between two toll booths when the driver was safely in their lane. The

segment of high stress situations were taken from the city drive where there was a busy traffic stretch.

The most efficient paper we referred to cites that electrodermal activity (EDA) and heart rate gives the best combination for stress detection. Here, we are making an attempt to constrict the signals to EDA by adding on some features so that an efficient algorithm can be proposed using one signal only. This would decrease the hardware requirements for the system, hence, the easiness of data collection increases and the system can be simplified.

The overview of proposed methodology has been shown in figure 1. We propose to use around 16 features derived from hand and foot galvanic skin response (GSR) as described in algorithm A. These include four statistical features, that are, mean and variance of normalized signals. The normalization of the signals is done using standard deviation. For normalization, the mean of the array of data values is first found out. Then, we find standard deviation. After that, all the values are normalised by subtracting mean from them and then dividing them by the deviation.



**Fig. 1: Overview of proposed methodology**

Eight features were calculated from skin conductivity signals to identify orienting responses. An orienting response is a hike in GSR signal due to activation of skin sweat glands as a consequence of sympathetic nervous activation. The proposed algorithm detects beginning and peaks of these responses by first extracting the slopes beyond a threshold value and then finding out the local minima just preceding it. Let the time duration between the onset and the peak be represented by  $O$  and the total increase in amplitude be represented by

Other than the aforementioned features, we also included two more features. One is the standard deviation of Power Spectral Density (PSD). Power spectral density gives us the power of a given signal in different frequency domains [9]. If we want to calculate the total power of signal in a frequency range, then, we need to take summation of PSD in that range. In this project, we calculated the standard deviation of PSD. Another feature is the total energy of the signal. When the driver is in high stress, then, the energy of the

signal will be more in comparison to when it is under low stress. These are calculated for both hand and foot GSR.

After obtaining a feature matrix of all the sample points, we used them to train two different classification algorithms and their testing results were compared. The algorithms are trained on all data points, except one, and then they are tested on that point. This is repeated for all data points in the sample space.

### **i) Algorithm A: Algorithm to create feature matrix**

*Input:*

*EDA signal of a driver for a duration of 5 minutes*

Steps:

begin

- 1) Identify all pairs of local minima and next subsequent maxima from the given signal.
- 2) Select pairs with slope more than the given threshold.
- 3) Count the number of such pairs.
- 4) Find the sum of difference in magnitude of minima and maxima pairs ( $\sum O_m$ ).
- 5) Find the sum of duration gap between minima and maxima pairs ( $\sum O_d$ ).
- 6) Calculate the summation of area under the minima-maxima pairs ( $\sum (1/2) \times O_m \times O_d$ ).
- 7) Determine mean and variance of the signal.
- 8) Calculate the total energy of the signal.
- 9) Determine PSD of given signal.
- 10) Calculate standard deviation of PSD.

end

*Output:*

*Feature matrix of given signal*

The Artificial Neural Network (ANN) was developed to imitate the human brain [10]. They are composed of multiple nodes, called neurons, which imitate the working of human neurons. Different neurons are connected by links. They take inputs, then perform some computation on those inputs and pass the results to other neurons. There are three layers of neurons in an ANN- input layer, hidden layer and output layer. The input layer is of the size of the feature matrix, i.e. 16; the output layer classifies each data point into low, medium and high stress levels, so it has 3 neurons; and the hidden layer consists of 10 neurons.

SVM classifies different classes via a hyperplane [11]. All the data points are plotted on a n-dimensional coordinate system and then a hyperplane separating them is identified. These planes will classify the data points in different classes.

Both ANN and SVM are used to obtain results by classifying test data and then their results are compared.

#### 4. RESULT AND DISCUSSION

We extracted sixteen features of 15 drivers, after segmenting 5 minute portions from the city and highway drive and the parts of absolute rest. Simulation was done using Python. Some examples of the feature values of foot GSR have been mentioned in table 1 and those of hand GSR have been mentioned in table 2.

i) **Table 1: Feature matrix of foot GSR**

S.No.	Mean	Varian ce	Freq. of Spike s	$\sum O_m$	$\sum O_d$	$\sum (1/2) \times O_m \times O_d$	Standard deviation of PSD	Total power of signal
1.	1.4714	0.4430	348	104.9041	677	35510.0279	27.14889	1681602.70716
2.	0.2149	3.0916	210	71.8662	1663	59756.7317	47721.492	13419906.9862
3.	1.0892	2.3309	227	55.0414	1347	37070.3890	1109.7485	10830122.1350

4.	1.8705	0.0074	114	7.4090	362	1341.0341	1.79164	1158609.76973
5.	1.6893	0.1293	143	92.4935	885	39540.9667	0.03740	875717.12470
6.	1.9540	0.0123	152	10.1547	581	2949.9232	1.53029	1272396.80090
7.	1.6382	0.0013	34	4.8343	745	1800.7573	5442.2257	173734503.298
8.	-0.2843	0.2189	151	28.4232	1714	24358.6722	367782.97	476634440.950
9.	1.4387	0.8624	260	16.8676	1693	14278.3785	33382.921	275625049.551

ii)

**Table 2: Feature matrix of hand GSR**

S.No.	Mean	Varian ce	Freq. of Spike s	$\sum O_m$	$\sum O_d$	$\sum (1/2) \times O_m \times O_d$	Standard deviation of PSD	Total power of signal
1.	2.4571	0.0113	176	18.1013	442	1296.9017	382.45555	29545830.263
2.	2.6775	0.0628	185	60.9350	1659	12595.0291	395131.40	431808220.57
3.	2.0521	0.0997	155	20.237	1435	4187.2150	54071.815	433201597.86
4.	-0.4281	0.2491	118	2.9883	868	428.1561	41.37305	4603272.1664
5.	0.9044	1.4615	165	25.8625	974	56736.2156	0.62812	2795189.3782
6.	1.6226	0.0445	109	7.8193	1071	25108.9256	48.26238	5093801.8501
7.	0.5014	0.1265	93	1.6218	528	4194.6209	714.63986	32491961.550
8.	0.5691	0.1802	179	65.8957	1722	29963.8783	59226.226	97886219.968
9.	0.6947	0.4610	141	30.5833	1642	11163.2520	3.26705	7733402.0151

When the feature matrices were passed into ANN, 71.11% accuracy was achieved as shown in table 3.

SVM provides better results than ANN as we achieved 86.66% accuracy using SVM. when we trained SVM on all data points except one and repeated this process for all points in sample space then the given efficiency was achieved. Hence, we can classify driver stress using only EDA by extracting its features and passing them in SVM.

iii)

**Table 3: Accuracy from different classifiers**

S. No.	Classifier	Accuracy
1.	Artificial Neural Network	71.11%
2.	Support Vector Machine	86.66%

## 5. SOURCE CODE

### i) Segmentation function(segmentation.py)

```
import numpy as np
mat=np.zeros(shape=(48,4651))
n=1;
for i in range(1,10):
    with open("/Users/vivek/Desktop/major/code/drive0" + i + ".mat") as file:
        valm=[[float(digit) for digit in line.split()] for line in file]

    val=np.array(valm)
    if( i!=2 and i!=3 and i!=4 and i!=1):
        col=6;
        mat[n,0:4650]= val[col,7440:12090]
        n=n+1;
    if(i==1):
        mat[n,0:4650]=val[col,16740:21390];
        n=n+1;
        mat[n,0:4650]=val[col,27900:32550];
        n=n+1;
```



```

elif(i==2 or i==4 or i==6 or i==5 or i==8 or i==9):
    mat[n,0:4650]=val[col,18600:23250];
    n=n+1;
    if(i==9):
        mat[n,0:4650]=val[col,32550:37200];
        n=n+1;
    else :
        mat[n,0:4650]=val[col,37200:41850];
        n=n+1;
elif(i==3 or i==7):
    mat[n,0:4650]=val[col,23250:27900];
    n=n+1;
    mat[n,0:4650]=val[col,41850:46500];
    n=n+1;
elif(i==2):
    col=4;
    mat[n,0:4650]= val[col,7440:12090];
    n=n+1;
    if(i==1):
        mat[n,0:4650]=val[col,16740:21390];
        n=n+1;
        mat[n,0:4650]=val[col,27900:32550];
        n=n+1;
elif(i==2 or i==4 or i==6 or i==5 or i==8 or i==9):
    mat[n,0:4650]=val[col,18600:23250];
    n=n+1;

```

```

if(i==9):
    mat[n,0:4650]=val[col,32550:37200];
    n=n+1;
else :
    mat[n,0:4650]=val[col,37200:41850];
    n=n+1;
elif (i==3 or i==7):
    mat[n,0:4650]=val[col,23250:27900];
    n=n+1;
    mat[n,0:4650]=val[col,41850:46500];
    n=n+1;
elif(i==4 or i==1):
    col=5;
    mat[n,0:4650]=val[col,7440:12090];
    n=n+1;
    if(i==1):
        mat[n,0:4650]=val[col,16740:21390];
        n=n+1;
        mat[n,0:4650]=val[col,27900:32550];
        n=n+1;
    elif(i==2 or i==4 or i==6 or i==5 or i==8 or i==9):
        mat[n,0:4650]=val[col,18600:23250];
        n=n+1;
        if(i==9):
            mat[n,0:4650]=val[col,32550:37200];
            n=n+1;

```

```

else :
    mat[n,0:4650]=val[col,37200:41850];
    n=n+1;
elif(i==3 or i==7):
    mat[n,0:4650]=val[col,23250:27900];
    n=n+1;
    mat[n,0:4650]=val[col,41850:46500];
    n=n+1;
elif(i==3):
    col=3;
    mat[n,0:4650]=val[col,7440:12090];
    n=n+1;
    if(i==1):
        mat[n,0:4650]=val[col,16740:21390];
        n=n+1;
        mat[n,0:4650]=val[col,27900:32550];
        n=n+1;
    elif(i==2 or i==4 or i==6 or i==5 or i==8 or i==9):
        mat[n,0:4650]=val[col,18600:23250];
        n=n+1;
        if(i==9):
            mat[n,0:4650]=val[col,32550:37200];
            n=n+1;
    else :
        mat[n,0:4650]=val[col,37200:41850];
        n=n+1;

```

```
elif (i==3 or i==7):
```

```
    mat[n,0:4650]=val[col,23250:27900];
```

```
    n=n+1;
```

```
    mat[n,0:4650]=val[col,41850:46500];
```

```
    n=n+1;
```

```
for i in range (10,17):
```

```
    with open("/Users/riyamaan/Desktop/major/code/drive0" + i + ".mat") as file:
```

```
        valm=[[float(digit) for digit in line.split()] for line in file]
```

```
val=np.array(valm)
```

```
if( i!=13 and i!=14):
```

```
    col=6;
```

```
    mat[n,0:4650]=val[col,7440:12090];
```

```
    n=n+1;
```

```
if(i==15 or i==16):
```

```
    mat[n,0:4650]=val[col,16740:21390];
```

```
    n=n+1;
```

```
    mat[n,0:4650]=val[col,27900:32550];
```

```
    n=n+1;
```

```
elif(i>=10 and i<=14):
```

```
    mat[n,0:4650]=val[col,18600:23250];
```

```
    n=n+1;
```

```
if(i==9):
```

```
    mat[n,0:4650]=val[col,32550:37200];
```

```
    n=n+1;
```

```

else :
    mat[n,0:4650]=val[col,37200:41850];
    n=n+1;
elif( i==13 or i==14):
    col=5;
    mat[n,0:4650]=val[col,7440:12090];
    n=n+1;
if(i==15 or i==16):
    mat[n,0:4650]=val[col,16740:21390];
    n=n+1;
    mat[n,0:4650]=val[col,27900:32550];
    n=n+1;
elif(i>=10 and i<=14):
    mat[n,0:4650]=val[col,18600:23250];
    n=n+1;
    if(i==9):
        mat[n,0:4650]=val[col,32550:37200];
        n=n+1;
    else :
        mat[n,0:4650]=val[col,37200:41850];
        n=n+1;

```

**ii) Function to calculate local maxima and minima (maxmin.py)**

```

import numpy as np
from peaks_func import peaks
def maxmin(arr):
    size=np.shape(arr)

```

```

row=int(size[0])
#col=int(size[1])
#temp=[]
#if(row>col):
#   for i in range(0,row):
#       temp.append(arr[i][0])
#   arr=temp
#size=np.shape(temp)
maxarr=[]
minarr=[]
for i in range(1,row-1):
    if(arr[i-1][0]<arr[i][0] and arr[i+1][0]<=arr[i][0]):
        if(arr[i+1][0]==arr[i][0]):
            j=i+1
            while(j+1<row-2 and arr[j+1][0]==arr[i][0]):
                j=j+1
            if(j+1<row-2 and arr[j+1][0]<arr[i][0]):
                maxarr.append(i)
        else:
            maxarr.append(i)
    if(arr[i-1][0]>=arr[i][0] and arr[i+1][0]>arr[i][0]):
        if(arr[i-1][0]==arr[i][0]):
            j=i-1
            while(arr[j-1][0]==arr[i][0] and j-1>0):
                j=j-1
            if(arr[j-1][0]>arr[i][0]):

```

```

        minarr.append(i)
    else:
        minarr.append(i)
size=np.shape(minarr)
sizemin=int(size[0])
size=np.shape(maxarr)
sizemax=int(size[0])
farr=[]
if(minarr[0]<maxarr[0] and sizemin>sizemax):
    for i in range(0,sizemin-1):
        farr.append(minarr[i])
        farr.append(maxarr[i])
elif (minarr[0]<maxarr[0] and sizemin==sizemax):
    for i in range(0,sizemin):
        farr.append(minarr[i])
        farr.append(maxarr[i])
elif (minarr[0]>maxarr[0] and sizemin==sizemax):
    for i in range(0,sizemax-1):
        farr.append(minarr[i])
        farr.append(maxarr[i+1])
elif (minarr[0]>maxarr[0] and sizemin<sizemax):
    for i in range(0,sizemax-1):
        farr.append(minarr[i])
        farr.append(maxarr[i+1])
return farr

```

**iii) Function to include peaks exceeding particular threshold  
(peaks\_func.py)**

```
import numpy as np

import math

def peaks(farr,iarr):

    size=np.shape(iarr)

    peaksarr=[]

    parr=[]

    maxs=90

    val=85

    for i in range(0,size[0]-1,2):

        slope=(farr[iarr[i+1]]-farr[iarr[i]])/(iarr[i+1]-iarr[i])

        slope=math.degrees(math.atan(slope))

        if(slope<val):

            parr.append(iarr[i])

            parr.append(iarr[i+1])

    i=0

    maxs=(farr[parr[i+1]]-farr[parr[i]])/(parr[i+1]-parr[i])

    maxs=math.degrees(math.atan(maxs))

    size=np.shape(parr)

    for i in range(2,size[0]-1,2):

        slope=(farr[parr[i+1]]-farr[parr[i]])/(parr[i+1]-parr[i])

        slope=math.degrees(math.atan(slope))
```



```

if(slope>maxs):
    maxs=slope

maxs=0.75*maxs

for i in range(0,size[0]-1,2):
    slope=(farr[parr[i+1]]-farr[parr[i]])/(parr[i+1]-parr[i])
    slope=math.degrees(math.atan(slope))
    if(slope>maxs):
        peaksarr.append(parr[i])
        peaksarr.append(parr[i+1])
return peaksarr

```

**iv) Function to normalise signal (normalise.py)**

```

import numpy as np

def normalise(arr):
    mean=0;
    size=np.shape(arr)
    for i in range(0,size[0]):
        mean+=arr[i]
    mean=mean/size[0]
    sqerror=0
    for i in range(0,size[0]):
        sqerror+=np.square((arr[i]-mean))
    dev=sqerror/size[0]
    dev=np.sqrt(dev)

```

```

#print(dev)

narr=np.zeros(shape=(size[0],1))

for i in range(0,size[0]):

    narr[i]=(arr[i]-mean)/dev

return narr

```

**v) Function to extract features (features\_func.py)**

```

import numpy as np

from normalise_func import normalise

from maxmin import maxmin

from peaks_func import peaks

from scipy import signal

def features(arr):          #no return is mentioned

    feat=np.zeros(shape=(1,8))    #8 features for the foot

    #power spectral density

    arr1=arr[1302:3349]

    (f,pxx)=signal.welch(arr1,15.5)

    feat[0][6]=np.std(pxx)

    #plt.semilogy(f, pxx)

    #print(pxx)

    #total power of signal

    power=0

    sizearr=np.shape(arr)

    for i in range(0,sizearr[0]):

        power+=pow(arr[i],2)

    power/=sizearr[0]

    feat[0][7]=power

```

```

size=np.shape(arr)
mean=0
for i in range(0,size[0]):
    mean+=arr[i]
mean=mean/size[0]
feat[0][4]=mean
var=0
for i in range(0,size[0]):
    var+=np.square(arr[i]-mean)
var=var/size[0]
feat[0][5]=var
arr=normalise(arr)                #normalize the signal
arr=arr.reshape((-1,1))          #second
iarr=maxmin(arr)                  #arr of max-min pairs
iarr=peaks(arr,iarr)  #peaks glitches are removed and 75% taken
size=np.shape(iarr)
num=size[0]/2
#print(num)
feat[0][0]=num
sum_min=0
sumd=0
area=0
for j in range(0,int(size[0]/2)):
    mag=(arr[iarr[(j*2)+1]]-arr[iarr[j*2]])
    sum_min+=mag
    dur=(iarr[(j*2)+1]-iarr[j*2])

```

```

    sumd+=dur
    area+=mag*dur/2

    #sum_min is summation of magnitudes
    #sumd is summation of duration between minima-maxima
    feat[0][1]=sum_min
    #print(sumd)
    feat[0][2]=sumd
    feat[0][3]=area
    #arr=normalise(arr)

    return feat

#array=foot[2]
#px=features(array)

```

**vi) Main function (main.py)**

```

import numpy as np

from features_func import features

with open('newfoot.txt') as file:

    footm=[[float(digit) for digit in line.split()] for line in file]

    foot=np.array(footm)

    size=np.shape(foot)

    row=int(size[0])

    col=int(size[1])

    feat_mat=np.zeros(shape=(row,16))

    for i in range(0,row):

        feat1=np.zeros(shape=(row,8))

        feat1=features(foot[i])

```

```

    feat_mat[i,:8]=feat1[:]
with open('newhand.txt') as file:
    handm=[[float(digit) for digit in line.split()] for line in file]
hand=np.array(handm)
size=np.shape(hand)
row=int(size[0])
col=int(size[1])
for i in range(0,row):
    feat1=np.zeros(shape=(row,8))
    feat1=features(hand[i])
    feat_mat[i,8:16]=feat1[:]
with open('feat_mat_peaks111.txt','wb') as file1:
    np.savetxt(file1,feat_mat, delimiter=',', fmt='%.5f')

```

#### vii) **Machine learning algorithms**

ANN and SVM are implemented using "sklearn" library of Python.

## **6. CONCLUSION**

In this project, we made an effort to classify automobile drivers stress into low, medium and high using only one physiological signal, that is EDA. It was collected from hand and foot of the driver in the Physionet drivers' stress detection dataset. Classification can be suitably done using SVM; since, it provides better results than ANN.

As our approach used only one signal, hence, the interference with driving is minimum as we do not need to collect more signals.

In future, some more EDA features can be found out, which can be helpful in enhancing the accuracy of the currently proposed system, so that, the stress of driver can be measured efficiently in real time. Future smart cars would include this system so as to make driving a relaxing experience for the driver and decrease the fatigue during highly stressed driving situations.

This project can be used in smart cars to be developed in future. We can use this technology to reduce by driver's stress after it has reached a certain threshold. This can be done by providing a relaxing environment within the car and diminishing the amount of information provided to the driver while driving. If the stress is very high, the driver can be asked to relax for some time before resuming driving.

## 7. PLAN OF WORK

Phases	Start Date-End Date	Work to be Done	Status
Dissertation Outline	17 Jan 2020 – 21 Jan 2020	Literature Review and prepare Dissertation Outline	Completed
Design & Development	22 Jan 2020 – 15 Feb 2020	Design & Development Activity	Completed
Testing	16 Feb 2020 – 13 Mar 2020	Software Testing, User Evaluation & Conclusion	Completed
Dissertation Review	14 Mar 2020-25 Mar 2020	Submit dissertation to Supervisor & Additional Examiner for review and feedback	Completed
Submission	26 Mar 2020-30 Mar 2020	Final Review and submission of dissertation	Completed

## 8. REFERENCES

- [1] Kopin, I. J., G. Eisenhofer, and D. Goldstein. "Sympathoadrenal medullary system and stress." *Mechanisms of physical and emotional stress*. Springer, Boston, MA, 1988. 11-23.
- [2] Goldberger, Ary L., et al. "PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals." *circulation* 101.23 (2000): e215-e220.
- [3] Lanatà, Antonio, et al. "How the autonomic nervous system and driving style change with incremental stressing conditions during simulated driving." *IEEE Transactions on Intelligent Transportation Systems* 16.3 (2014): 1505-1517.
- [4] Zheng, Rencheng, et al. "Biosignal analysis to assess mental stress in automatic driving of trucks: Palmar perspiration and masseter electromyography." *Sensors* 15.3 (2015): 5136-5150.
- [5] Lee, Boon-Giin, and Wan-Young Chung. "Wearable glove-type driver stress detection using a motion sensor." *IEEE Transactions on Intelligent Transportation Systems* 18.7 (2016): 1835-1844.
- [6] Lee, Dae Seok, Teak Wei Chong, and Boon Giin Lee. "Stress events detection of driver by wearable glove system." *IEEE Sensors Journal* 17.1 (2016): 194-204.
- [7] Singh, Rajiv Ranjan, Sailesh Conjeti, and Rahul Banerjee. "Biosignal based on-road stress monitoring for automotive drivers." *2012 National Conference on Communications (NCC)*. IEEE, 2012.
- [8] Healey, Jennifer A., and Rosalind W. Picard. "Detecting stress during real-world driving tasks using physiological sensors." *IEEE Transactions on intelligent transportation systems* 6.2 (2005): 156-166.
- [9] Parhi, Keshab K., and Manohar Ayinala. "Low-complexity Welch power spectral density computation." *IEEE Transactions on Circuits and Systems I: Regular Papers* 61.1 (2013): 172-182.
- [10] *Neural network models (supervised) — scikit-learn 0.18.1 documentation*. [Online]. Available: [http://scikit-learn.org/stable/modules/neural\\_networks\\_supervised.html](http://scikit-learn.org/stable/modules/neural_networks_supervised.html). [Accessed: 20-Apr-2017].
- [11] *Support Vector Machines — scikit-learn 0.18.1 documentation*. [Online]. Available: <http://scikit-learn.org/stable/modules/svm.html>. [Accessed: 20-Apr-2017].

