Downloading Train and Test data

In [1]:

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

In [ ]:

```
import os
os.environ['KAGGLE_CONFIG_DIR'] = "/content/gdrive/My Drive"
%cd /content/gdrive/My Drive/
```

```
/content/gdrive/My Drive
```

In [34]:

```
import os
os.environ['KAGGLE_CONFIG_DIR'] = "/content"
%cd /content
```

```
/content
```

In [30]:

```
pip install kaggle --upgrade
```

```
Requirement already up-to-date: kaggle in /usr/local/lib/python3.6/dist-packages (1.5.8)
Requirement already satisfied, skipping upgrade: python-slugify in /usr/local/lib/python3.6/dist-p
ackages (from kaggle) (4.0.1)
Requirement already satisfied, skipping upgrade: requests in /usr/local/lib/python3.6/dist-
packages (from kaggle) (2.23.0)
Requirement already satisfied, skipping upgrade: slugify in /usr/local/lib/python3.6/dist-packages
(from kaggle) (0.0.1)
Requirement already satisfied, skipping upgrade: python-dateutil in /usr/local/lib/python3.6/dist-
packages (from kaggle) (2.8.1)
Requirement already satisfied, skipping upgrade: six>=1.10 in /usr/local/lib/python3.6/dist-
packages (from kaggle) (1.15.0)
Requirement already satisfied, skipping upgrade: certifi in /usr/local/lib/python3.6/dist-packages
(from kaggle) (2020.6.20)
Requirement already satisfied, skipping upgrade: tqdm in /usr/local/lib/python3.6/dist-packages (f
rom kaggle) (4.41.1)
Requirement already satisfied, skipping upgrade: urllib3<1.25,>=1.21.1 in
/usr/local/lib/python3.6/dist-packages (from kaggle) (1.24.3)
Requirement already satisfied, skipping upgrade: text-unidecode>=1.3 in
/usr/local/lib/python3.6/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied, skipping upgrade: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-pac
kages (from requests->kaggle) (2.10)
Requirement already satisfied, skipping upgrade: chardet<4,>=3.0.2 in
/usr/local/lib/python3.6/dist-packages (from requests->kaggle) (3.0.4)
```

In [7]:

```
!kaggle competitions download -c reducing-commercial-aviation-fatalities
```

```
Warning: Looks like you're using an outdated API Version, please consider updating (server 1.5.6 /
client 1.5.4)
Downloading test.csv.zip to /content/gdrive/My Drive
 99% 1.66G/1.67G [01:04<00:00, 38.1MB/s]
100% 1.67G/1.67G [01:05<00:00, 27.5MB/s]
sample_submission.csv.zip: Skipping, found more recently modified local copy (use --force to force
download)
Downloading train.csv.zip to /content/gdrive/My Drive
 99% 433M/435M [00:20<00:00, 26.5MB/s]
100% 435M/435M [00:20<00:00, 22.5MB/s]
```

In [ ]:

```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.135 Safari/537.36" --hea
der="Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/s
d-exchange;v=b3;q=0.9" --header="Accept-Language: en,en-US;q=0.9,fr;q=0.8" --header="Referer: http
s://www.kaggle.com/" "https://storage.googleapis.com/kaggle-competitions-data/kaggle-
v2/11835/224935/compressed/train.csv.zip?GoogleAccessId=web-data@kaggle-
161607.iam.gserviceaccount.com&Expires=1599644361&Signature=LlvOjsVvmysGbPCsoUt88wGFZY3%2BfbhktYPFw
STAdkoY2X9a%2Bz84KzhMJG95LaJ2SYDV2BZqJ4FYnSQKu9c9yqTBvW0fuU2f8SwG%2F7I0yJ3m%2BX6PABbFV4Rp7qowpWIscX
wCa6yZygYOetmBs2LwBCMgts9bKd8ArM5i3cgGc5S7akD4J8lFVjmbZEA3QOfSeUhoVPZN1BHE49DgS2FwqLzmihXtlFCjQFyEC
rEGfZBJfyi%2Fhu3bfk8IBf3CXZgmmAhH%2ByyFTQJPOPDuJiqBi9Wa0AhaVzBVkLwhI6NYWAgQulSLXOOfsoDSxxpNGRkx2ync
Rg%3D%3D&response-content-disposition=attachment%3B+filename%3Dtrain.csv.zip" -c -O
'train.csv.zip'
```

```
--2020-09-12 07:20:58--  https://storage.googleapis.com/kaggle-competitions-data/kaggle-
v2/11835/224935/compressed/train.csv.zip?GoogleAccessId=web-data@kaggle-
161607.iam.gserviceaccount.com&Expires=1599644361&Signature=LlvOjsVvmysGbPCsoUt88wGFZY3%2BfbhktYPFw
STAdkoY2X9a%2Bz84KzhMJG95LaJ2SYDV2BZqJ4FYnSQKu9c9yqTBvW0fuU2f8SwG%2F7I0yJ3m%2BX6PABbFV4Rp7qowpWIscX
wCa6yZygYOetmBs2LwBCMgts9bKd8ArM5i3cgGc5S7akD4J8lFVjmbZEA3QOfSeUhoVPZN1BHE49DgS2FwqLzmihXtlFCjQFyEC
rEGfZBJfyi%2Fhu3bfk8IBf3CXZgmmAhH%2ByyFTQJPOPDuJiqBi9Wa0AhaVzBVkLwhI6NYWAgQulSLXOOfsoDSxxpNGRkx2ync
Rg%3D%3D&response-content-disposition=attachment%3B+filename%3Dtrain.csv.zip
Resolving storage.googleapis.com (storage.googleapis.com)... 108.177.126.128, 74.125.128.128, 173.
194.69.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|108.177.126.128|:443... connected.
HTTP request sent, awaiting response... 400 Bad Request
2020-09-12 07:20:58 ERROR 400: Bad Request.
```

In [ ]:

```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.135 Safari/537.36" --hea
der="Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/s
d-exchange;v=b3;q=0.9" --header="Accept-Language: en,en-US;q=0.9,fr;q=0.8" --header="Referer: http
s://www.kaggle.com/" "https://storage.googleapis.com/kaggle-competitions-data/kaggle-
v2/11835/224935/compressed/test.csv.zip?GoogleAccessId=web-data@kaggle-
161607.iam.gserviceaccount.com&Expires=1599644475&Signature=bYW9J4eC8YXde8Bm%2F4rF8MBzZ5W0DOlRuZfVf
7fu%2B2JE%2B7dNAKxslsJUIVSLFDqdJjL2ACOZhxRIkTfSrX8Auuyxb66VCXgbqJcs5dD7SRnEkqhDBJdx%2F3p0rFWn2VupvH
efTjjo4SJQsG5Wkd%2BmCJR%2FT25WVJ6aUapxkT6YoaCKt9mJMvOP0QGQ9YmsO0HUZ58qy7OpBeezithU50OUSpsEwFL%2FxyI
%2FeCKGZjws9UJeCRU17iVzOxQMtN1mJ%2Bqk6NbkL%2Bxo9wZe2BHPWA5EsvUQmNLR%2Bg2Dn64Ri1HyL6njZg44CqdGVo8fhN
Ero2%2FII0TKrg%3D%3D&response-content-disposition=attachment%3B+filename%3Dtest.csv.zip" -c -O 'te
st.csv.zip'
```

```
--2020-09-11 14:34:56--  https://storage.googleapis.com/kaggle-competitions-data/kaggle-
v2/11835/224935/compressed/test.csv.zip?GoogleAccessId=web-data@kaggle-
161607.iam.gserviceaccount.com&Expires=1599644475&Signature=bYW9J4eC8YXde8Bm%2F4rF8MBzZ5W0DOlRuZfVf
7fu%2B2JE%2B7dNAKxslsJUIVSLFDqdJjL2ACOZhxRIkTfSrX8Auuyxb66VCXgbqJcs5dD7SRnEkqhDBJdx%2F3p0rFWn2VupvH
efTjjo4SJQsG5Wkd%2BmCJR%2FT25WVJ6aUapxkT6YoaCKt9mJMvOP0QGQ9YmsO0HUZ58qy7OpBeezithU50OUSpsEwFL%2FxyI
%2FeCKGZjws9UJeCRU17iVzOxQMtN1mJ%2Bqk6NbkL%2Bxo9wZe2BHPWA5EsvUQmNLR%2Bg2Dn64Ri1HyL6njZg44CqdGVo8fhN
Ero2%2FII0TKrg%3D%3D&response-content-disposition=attachment%3B+filename%3Dtest.csv.zip
Resolving storage.googleapis.com (storage.googleapis.com)... 64.233.189.128, 108.177.97.128,
108.177.125.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|64.233.189.128|:443... connected.
HTTP request sent, awaiting response... 400 Bad Request
2020-09-11 14:34:57 ERROR 400: Bad Request.
```

In [2]:

```
!pip install phik
```

```
Collecting phik
  Downloading
https://files.pythonhosted.org/packages/01/5a/7ef1c04ce62cd72f900c06298dc2385840550d5c653a0dbc19109
7e6/phik-0.10.0-py3-none-any.whl (599kB)
     |████████████████████████████████| 604kB 3.5MB/s
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.6/dist-packages (from phik)
(1.4.1)
```

```
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.6/dist-packages (from phik)
(1.18.5)
Requirement already satisfied: numba>=0.38.1 in /usr/local/lib/python3.6/dist-packages (from phik)
(0.48.0)
Requirement already satisfied: matplotlib>=2.2.3 in /usr/local/lib/python3.6/dist-packages (from
phik) (3.2.2)
Requirement already satisfied: joblib>=0.14.1 in /usr/local/lib/python3.6/dist-packages (from
phik) (0.16.0)
Requirement already satisfied: pandas>=0.23.4 in /usr/local/lib/python3.6/dist-packages (from
phik) (1.0.5)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from
numba>=0.38.1->phik) (50.3.0)
Requirement already satisfied: llvmlite<0.32.0,>=0.31.0dev0 in /usr/local/lib/python3.6/dist-
packages (from numba>=0.38.1->phik) (0.31.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from
matplotlib>=2.2.3->phik) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from
matplotlib>=2.2.3->phik) (1.2.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages
(from matplotlib>=2.2.3->phik) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.6/dist-packages (from matplotlib>=2.2.3->phik) (2.4.7)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from
pandas>=0.23.4->phik) (2018.9)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from cycler>=0.10-
>matplotlib>=2.2.3->phik) (1.15.0)
Installing collected packages: phik
Successfully installed phik-0.10.0
```

In [48]:

```
from google.colab import files
files.upload()
```

Choose File No file selected

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle (1).json
```

Out[48]:

```
{'kaggle.json': b'{"username":"sankalpchawla","key":"1d2fc96a3f385db0c3807663a6ffb2c5"}'}
```

In [52]:

```
!pip install -q kaggle
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!ls ~/.kaggle
!chmod 600 /root/.kaggle/kaggle.json
```

```
kaggle.json
```

Unzipping the files

In [3]:

```
!unzip -q "/content/gdrive/My Drive/train.csv.zip" -d "/content/"
```

In [4]:

```
!unzip -q "/content/gdrive/My Drive/test.csv.zip" -d "/content/"
```

Importing required packages

In [5]:

```
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
import phik
from phik import resources, report
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import TSNE
from scipy import stats
import imblearn
import warnings
import math
import pickle
from sklearn.externals import joblib
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import log_loss
from prettytable import PrettyTable
import lightgbm as lgb
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
warnings.filterwarnings('ignore')
from sklearn.preprocessing import MinMaxScaler
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning:
pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
/usr/local/lib/python3.6/dist-packages/sklearn/externals/six.py:31: FutureWarning: The module is d
eprecated in version 0.21 and will be removed in version 0.23 since we've dropped support for
Python 2.7. Please rely on the official version of six (https://pypi.org/project/six/).
  "(https://pypi.org/project/six/).", FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:144: FutureWarning: The
sklearn.neighbors.base module is  deprecated in version 0.22 and will be removed in version 0.24.
The corresponding classes / functions should instead be imported from sklearn.neighbors. Anything
that cannot be imported from sklearn.neighbors is now part of the private API.
  warnings.warn(message, FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/externals/joblib/__init__.py:15: FutureWarning: skl
earn.externals.joblib is deprecated in 0.21 and will be removed in 0.23. Please import this
functionality directly from joblib, which can be installed with: pip install joblib. If this
warning is raised when loading pickled models, you may need to re-serialize those models with scik
it-learn 0.21+.
  warnings.warn(msg, category=FutureWarning)
```

Exploratory Data Analysis

In [6]:

```
df_train = pd.read_csv('/content/train.csv')
df_test = pd.read_csv('/content/test.csv')
```

Getting initial impression of data and columns

In [ ]:

```
df_train.head()
```

Out[ ]:

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2
0	1	CA	0.011719	1	-5.28545	26.775801	-9.527310	-12.793200	16.717800	33.737499	23.712299	-6.695870
1	1	CA	0.015625	1	-2.42842	28.430901	-9.323510	-3.757230	15.969300	30.443600	21.010300	-6.474720

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2
2	1	CA	0.019531	1	10.67150	30.420200	15.350700	24.724001	16.143101	32.142799	25.431801	0.088707
3	1	CA	0.023438	1	11.45250	25.609800	2.433080	12.412500	20.533300	31.494101	19.142799	-0.256516
4	1	CA	0.027344	1	7.28321	25.942600	0.113564	5.748000	19.833599	28.753599	20.572100	-1.953470

In [ ]:

```
df_train.tail()
```

Out[ ]:

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	
4867416	13	SS	99.991005	1	-47.758202	-71.050400	-37.980801	-33.852901	-59.413601	-21.614300	-26.437300	-4
4867417	13	SS	99.993004	0	-24.536699	79.787399	-45.435699	-6.329290	25.052500	-13.852700	23.737301	-7
4867418	13	SS	99.994003	1	-42.078499	-66.937401	-27.298201	-25.931900	-47.430302	-17.464199	-20.454201	-3
4867419	13	SS	99.997002	0	20.536301	10.485500	63.723400	44.422600	95.535500	76.922096	-65.109299	4
4867420	13	SS	99.998001	1	-12.230700	-39.962601	-2.675310	5.699130	-22.091000	-7.601820	6.032570	-

In [ ]:

```
df_test.head()
```

Out[ ]:

	id	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	ee
0	0	1	LOFT	0.000000	0	17.899500	6.127830	0.994807	-28.206200	-47.695499	-187.080002	-33.183498	-4.22
1	1	1	LOFT	0.000000	1	45.883202	94.749001	23.290800	1.392000	2.060940	-5.145290	6.395940	33.4
2	2	1	LOFT	0.003906	0	33.120098	28.356501	-7.239220	-7.690860	-25.833799	-107.236000	12.845200	1.21
3	3	1	LOFT	0.003906	1	43.280102	95.887001	18.702299	-1.432890	-4.232600	-8.021180	7.427430	27.3
4	4	1	LOFT	0.007812	0	7.929110	3.460380	-10.860800	-26.366699	-25.894699	37.007900	-50.334202	-11.6

In [ ]:

```
df_test.tail()
```

Out[ ]:

	id	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5
17965138	17965138	13	LOFT	999.988281	1	14.516900	14.190900	11.963100	4.805490	4.343070	9.962720
17965139	17965139	13	LOFT	999.992188	0	58.374599	75.113197	67.583298	67.535797	84.844002	30.198299
17965140	17965140	13	LOFT	999.992188	1	7.778510	13.460200	5.681790	-5.885640	0.586223	36.501202

	id	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5
17965141	17965141	13	LOFT	999.996094	0	64.646202	79.033997	54.108002	49.453701	22.497400	38.881302
17965142	17965142	13	LOFT	999.996094	1	7.645440	11.739200	4.093190	-5.170130	4.204370	31.264200

In [ ]:

```
print('Shape of training data: ', df_train.shape)
```

```
Shape of training data:  (4867421, 28)
```

In [ ]:

```
print('Shape of test data: ', df_test.shape)
```

```
Shape of test data:  (17965143, 28)
```

Checking datatypes and null/missing values in all the columns

In [ ]:

```
df_train.info(verbose=True,null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4867421 entries, 0 to 4867420
Data columns (total 28 columns):
 #   Column      Non-Null Count    Dtype  
---  ------      --------------    -----  
 0   crew        4867421 non-null  int64  
 1   experiment  4867421 non-null  object 
 2   time        4867421 non-null  float64
 3   seat        4867421 non-null  int64  
 4   eeg_fp1     4867421 non-null  float64
 5   eeg_f7      4867421 non-null  float64
 6   eeg_f8      4867421 non-null  float64
 7   eeg_t4      4867421 non-null  float64
 8   eeg_t6      4867421 non-null  float64
 9   eeg_t5      4867421 non-null  float64
 10  eeg_t3      4867421 non-null  float64
 11  eeg_fp2     4867421 non-null  float64
 12  eeg_o1      4867421 non-null  float64
 13  eeg_p3      4867421 non-null  float64
 14  eeg_pz      4867421 non-null  float64
 15  eeg_f3      4867421 non-null  float64
 16  eeg_fz      4867421 non-null  float64
 17  eeg_f4      4867421 non-null  float64
 18  eeg_c4      4867421 non-null  float64
 19  eeg_p4      4867421 non-null  float64
 20  eeg_poz     4867421 non-null  float64
 21  eeg_c3      4867421 non-null  float64
 22  eeg_cz      4867421 non-null  float64
 23  eeg_o2      4867421 non-null  float64
 24  ecg         4867421 non-null  float64
 25  r           4867421 non-null  float64
 26  gsr         4867421 non-null  float64
 27  event       4867421 non-null  object 
dtypes: float64(24), int64(2), object(2)
memory usage: 1.0+ GB
```

In [ ]:

```
df_test.info(verbose=True,null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17965143 entries, 0 to 17965142
Data columns (total 28 columns):
 #   Column      Non-Null Count     Dtype  
---  ------      --------------     -----  
 0   id          17965143 non-null  int64
```

```
 1   crew         17965143 non-null  int64
 2   experiment   17965143 non-null  object
 3   time         17965143 non-null  float64
 4   seat         17965143 non-null  int64
 5   eeg_fp1      17965143 non-null  float64
 6   eeg_f7       17965143 non-null  float64
 7   eeg_f8       17965143 non-null  float64
 8   eeg_t4       17965143 non-null  float64
 9   eeg_t6       17965143 non-null  float64
 10  eeg_t5       17965143 non-null  float64
 11  eeg_t3       17965143 non-null  float64
 12  eeg_fp2      17965143 non-null  float64
 13  eeg_o1       17965143 non-null  float64
 14  eeg_p3       17965143 non-null  float64
 15  eeg_pz       17965143 non-null  float64
 16  eeg_f3       17965143 non-null  float64
 17  eeg_fz       17965143 non-null  float64
 18  eeg_f4       17965143 non-null  float64
 19  eeg_c4       17965143 non-null  float64
 20  eeg_p4       17965143 non-null  float64
 21  eeg_poz      17965143 non-null  float64
 22  eeg_c3       17965143 non-null  float64
 23  eeg_cz       17965143 non-null  float64
 24  eeg_o2       17965143 non-null  float64
 25  ecg          17965143 non-null  float64
 26  r            17965143 non-null  float64
 27  gsr          17965143 non-null  float64
dtypes: float64(24), int64(3), object(1)
memory usage: 3.7+ GB
```

Observations: No missing or null value is present in the train and test dataset.

Check statistical summary of dataset

In [ ]:

```
df_train.describe()
```

Out[ ]:

	crew	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6
count	4.867421e+06	4.867421e+06	4.867421e+06	4.867421e+06	4.867421e+06	4.867421e+06	4.867421e+06	4.867421e+06
mean	5.538783e+00	1.782358e+02	4.999531e-01	3.746336e+00	1.360002e+00	1.213644e+00	7.350926e-02	7.845481e-02
std	3.409353e+00	1.039592e+02	5.000000e-01	4.506763e+01	3.518923e+01	3.519242e+01	2.431472e+01	1.803932e+01
min	1.000000e+00	3.000000e-03	0.000000e+00	-1.361360e+03	-1.581330e+03	-1.643950e+03	-1.516640e+03	-1.220510e+03
25%	3.000000e+00	8.808100e+01	0.000000e+00	-9.200250e+00	-8.325150e+00	-8.767610e+00	-7.367240e+00	-6.102000e+00
50%	5.000000e+00	1.769297e+02	0.000000e+00	3.819020e-01	4.264100e-02	1.140390e-01	0.000000e+00	0.000000e+00
75%	7.000000e+00	2.683398e+02	1.000000e+00	1.030610e+01	8.753340e+00	9.282560e+00	7.437780e+00	6.176630e+00
max	1.300000e+01	3.603711e+02	1.000000e+00	1.972240e+03	2.048790e+03	2.145710e+03	1.731880e+03	9.009370e+02

In [ ]:

```
df_test.describe()
```

Out[ ]:

	id	crew	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4
count	1.796514e+07	1.796514e+07	1.796514e+07	1.796514e+07	1.796514e+07	1.796514e+07	1.796514e+07	1.796514e+07
mean	8.982571e+06	5.240582e+00	1.972621e+03	4.999695e-01	5.938414e+00	2.650973e+00	1.394608e+00	-2.414278e-01
std	5.186090e+06	3.408746e+00	1.168732e+03	5.000000e-01	1.838633e+02	1.746916e+02	1.739371e+02	1.652322e+02

	id	crew	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4
min	0.000000e+00	1.000000e+00	0.000000e+00	0.000000e+00	-4.629900e+03	-4.635800e+03	-4.629850e+03	-4.632290e+03
25%	4.491286e+06	2.000000e+00	9.746914e+02	0.000000e+00	-2.105640e+01	-1.958760e+01	-1.974340e+01	-1.629735e+01
50%	8.982571e+06	5.000000e+00	1.949363e+03	0.000000e+00	8.436010e-01	2.468260e-01	4.586300e-01	1.185930e-01
75%	1.347386e+07	7.000000e+00	2.924035e+03	1.000000e+00	2.475940e+01	2.129610e+01	2.182585e+01	1.684330e+01
max	1.796514e+07	1.300000e+01	4.917688e+03	1.000000e+00	4.751060e+03	4.637460e+03	4.756000e+03	4.729170e+03

Observations:

- From inital look at the data we can see that the range of mean,std,all quantiles range values and maximum values varies in test data as compared to train data. So we can say that the distribution of the data is more varied in test data.
- The mean value and the median value in most columns don't have much difference but in some columns this differnece is large so that maybe be probably due to outliers.
- Further plotting of the data will give us more clear picture.

Some basic checks on the dataset

In [ ]:

```
df_train.loc[(df_train['experiment'] == 'SS') & (df_train['event'] == 'A')]
```

Out[ ]:

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	
368506	1	SS	0.011719	1	-0.422763	0.169267	7.011450	-16.576401	-0.862656	-9.070770	-8.478650	-
368507	1	SS	0.015625	1	-0.591358	-4.215710	14.462300	-7.315730	-2.807000	-10.119500	-8.585020	-
368508	1	SS	0.019531	1	0.111509	-8.353730	13.947000	7.367510	-8.511370	-3.220590	-7.488520	-
368509	1	SS	0.023438	1	5.002080	-6.760340	14.993200	-5.567270	1.916160	1.011310	-5.673370	4
368510	1	SS	0.027344	1	16.250401	10.983400	23.974600	6.905360	1.771920	3.632250	7.542810	1
...	...	...	...	...	...	...	...	...	...	...	...	.
4867416	13	SS	99.991005	1	-47.758202	-71.050400	-37.980801	-33.852901	-59.413601	-21.614300	-26.437300	-
4867417	13	SS	99.993004	0	-24.536699	79.787399	-45.435699	-6.329290	25.052500	-13.852700	23.737301	-
4867418	13	SS	99.994003	1	-42.078499	-66.937401	-27.298201	-25.931900	-47.430302	-17.464199	-20.454201	-
4867419	13	SS	99.997002	0	20.536301	10.485500	63.723400	44.422600	95.535500	76.922096	-65.109299	4
4867420	13	SS	99.998001	1	-12.230700	-39.962601	-2.675310	5.699130	-22.091000	-7.601820	6.032570	-

1420055 rows × 28 columns

In [ ]:

```
df_train.experiment.unique()
```

Out[ ]:

```
array(['CA', 'DA', 'SS'], dtype=object)
```

In [ ]:

```
df_train.event.unique()
```

Out[ ]:

```
array(['A', 'C', 'D', 'B'], dtype=object)
```

In [ ]:

```
df_test.experiment.unique()
```

Out[ ]:

```
array(['LOFT'], dtype=object)
```

Observations:

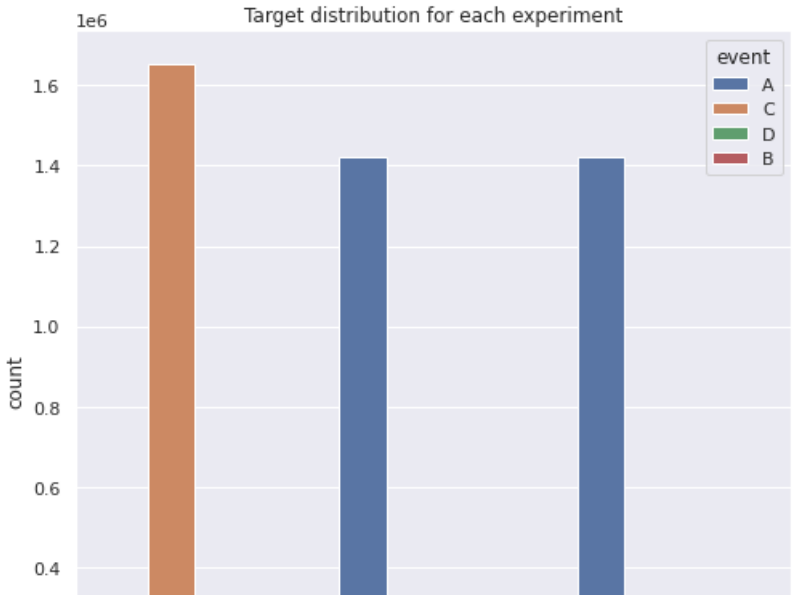
- From data we can see the experiment is one of CA, DA, SS or LOFT. The first 3 comprise the training set. The latter the test set.
- Event is state of the pilot at the given time: one of A = baseline, B = SS, C = CA, D = DA
- Since we are predicting the event so that column is not present in the test data.
- The training set is comprised of a set of controlled experiments collected in a non-flight environment, outside of a flight simulator.
- The test set (abbreviated LOFT = Line Oriented Flight Training) consists of a full flight (take off, flight, and landing) in a flight simulator.
- The training set contains three experiments (one for each state) in which the pilots experienced just one of the states. For example, in the experiment = CA, the pilots were either in a baseline state (no event) or the CA state. Similarly for other experiments as well either they are in baseline state or any of rest of the 3.

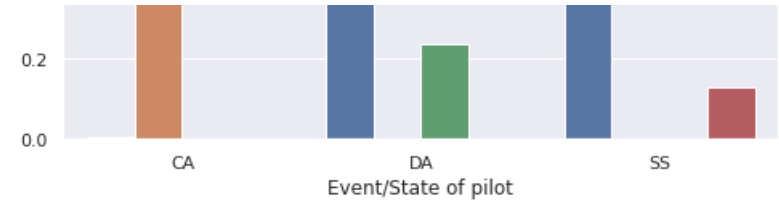
General analysis of all the features

Column - Experiment and Event

In [ ]:

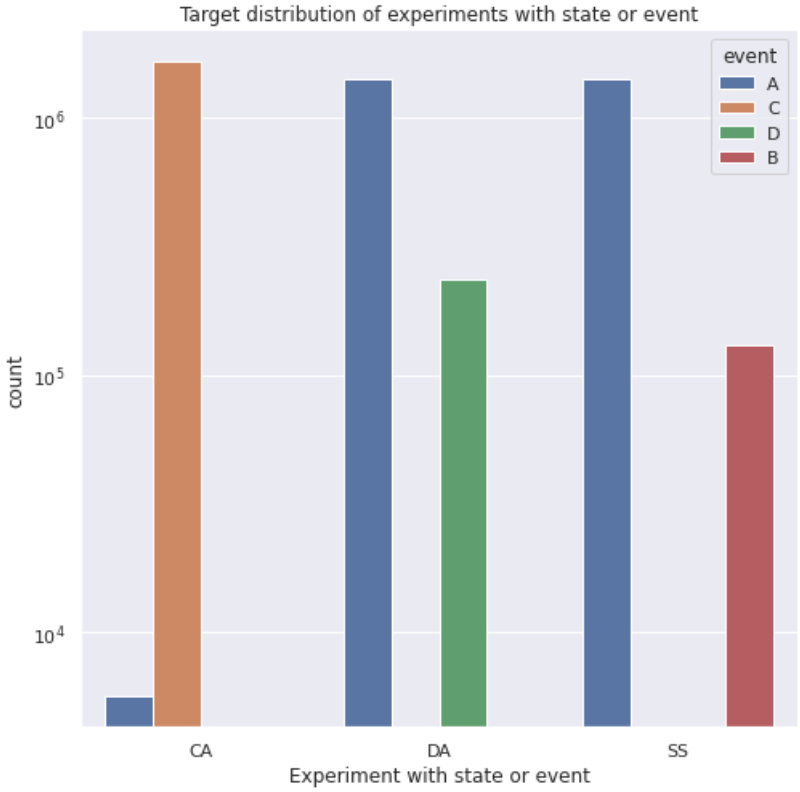
```
#https://seaborn.pydata.org/generated/seaborn.countplot.html
#https://www.geeksforgeeks.org/countplot-using-seaborn-in-python/
plt.figure(figsize=(8,8))
sns.set(style="darkgrid")
ax = sns.countplot(x='experiment', hue='event' ,data=df_train)
plt.xlabel('Event/State of pilot')
plt.title('Target distribution for each experiment')
plt.show()
```





In [ ]:

```
plt.figure(figsize=(8,8))
sns.set(style="darkgrid")
ax = sns.countplot(x='experiment',hue='event', data=df_train)
plt.yscale('log')
plt.xlabel('Experiment with state or event')
plt.title('Target distribution of experiments with state or event')
plt.show()
```



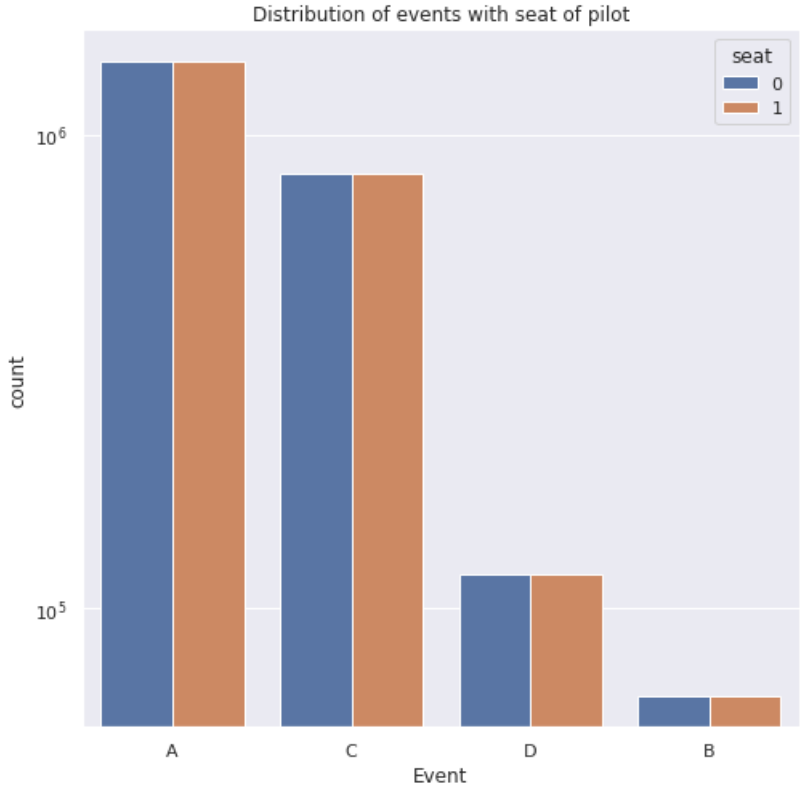
Observations:

- From first plot we can see that train data is highly imbalanced as event A occurs far more time than any other event.
- In second plot log is taken on scale of y axis to fit all the data. Further we can observe that for the experiment CA pilots are in distracted state as compared to baseline state.
- In other 2 experiments in second plot the pilots are more in base line state as compared to distracted DA or SS state.
- Number of experiments for DA and SS are also more than the number of experiments for CA.
- But experiment CA is responsible for most number of distracted states of pilot in dataset.
- So from this we can conclude that experiment CA is most likely to put the pilot in distracted state as compared to other experiments.

Column - Seat and Crew

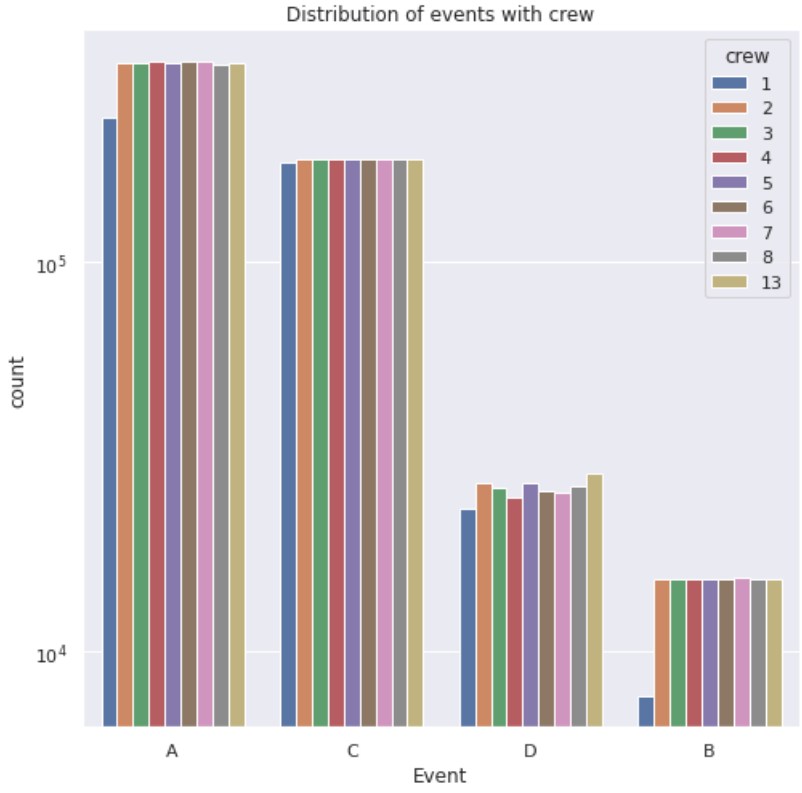
In [ ]:

```
plt.figure(figsize=(8,8))
sns.set(style="darkgrid")
ax = sns.countplot(x='event',hue='seat', data=df_train)
plt.yscale('log')
plt.xlabel('Event')
plt.title('Distribution of events with seat of pilot')
plt.show()
```



In [ ]:

```
plt.figure(figsize=(8,8))
sns.set(style="darkgrid")
ax = sns.countplot(x='event',hue='crew', data=df_train)
plt.yscale('log')
plt.xlabel('Event')
plt.title('Distribution of events with crew')
plt.show()
```



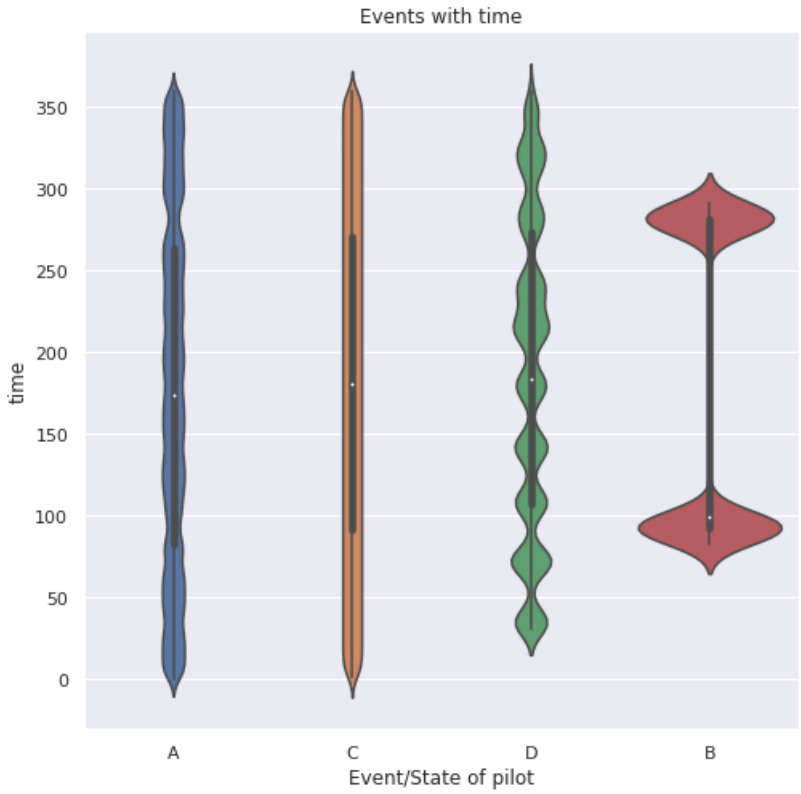
Observations:

- For every event outcome number of experiments on left and right seat are equal.
- Moreover seat cannot play any role in prediction as state of mind is independent from the side of seat where a person is sitting.
- The type of experiment can be a useful feature.

Column - Time

In [ ]:

```
plt.figure(figsize=(8,8))
sns.set(style="darkgrid")
ax = sns.violinplot(x="event", y="time", data=df_train)
plt.xlabel('Event/State of pilot')
plt.title('Events with time')
plt.show()
```



Observations:

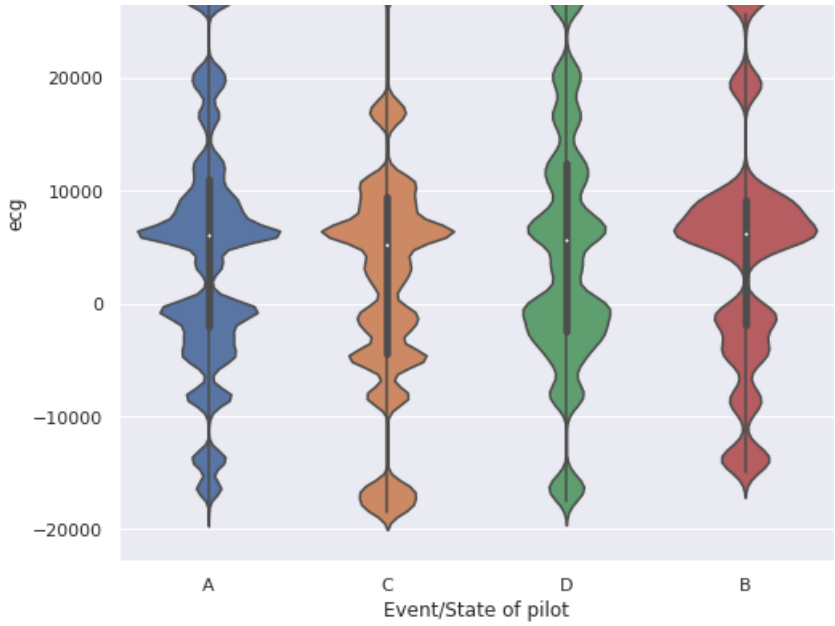
- We can see some different patterns emerging from time distribution which will be studied further in time series analysis of data and they can be useful in predicting results.
- The event B is occuring at low and high range in all the distributions and event D has a alternating distribution

Column - ECG (Electrocardiogram)

- The sensor had a resolution/bit of .012215 μV and a range of -100mV to +100mV. The data are provided in microvolts.

In [ ]:

```
plt.figure(figsize=(8,8))
sns.set(style="darkgrid")
ax = sns.violinplot(x="event", y="ecg", data=df_train)
plt.xlabel('Event/State of pilot')
plt.title('Events with ECG')
plt.show()
```

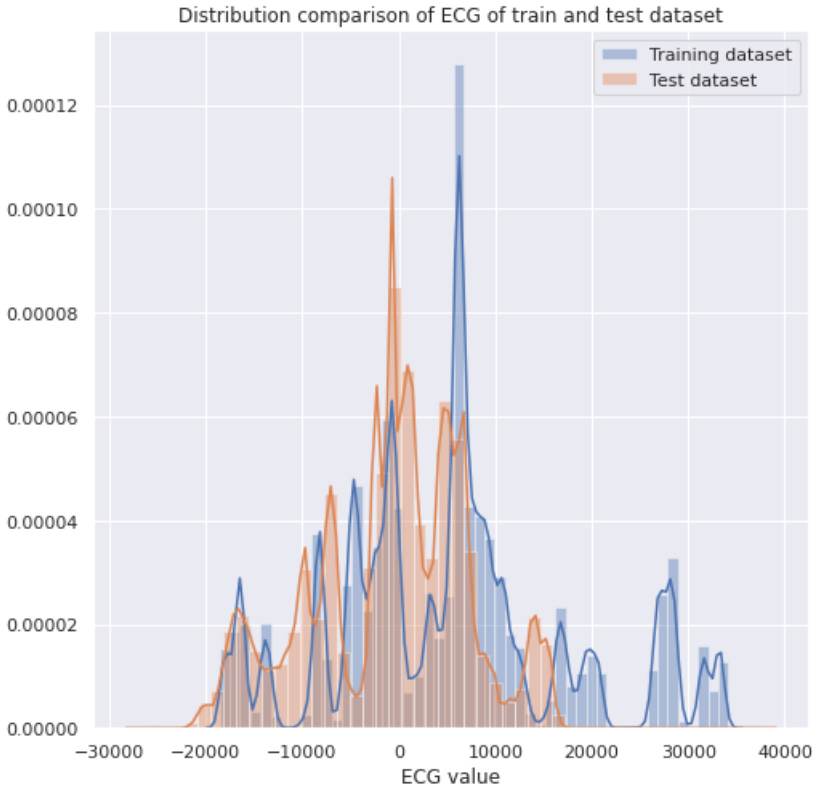


Observations:

- We can see from distribution that there are areas where there are almost very few or none values of ecg for the event. Many of the parts between pair of events are overlapping but the range of values vary vastly.So, it could be a good feature
- The values in extreme low and higher ranges cannot be classified as outliers as the density of the values is quite good.
- This feature could be useful as for a range of values it will help to narrow down the state.
- For event A and B the maximum distribution concentration is between 20k uV to -10k uV. For event D it is bit more evenly distributed and for event B concentration is higher in 10k uV to the lower ranges of -20k uV.

In [ ]:

```
plt.figure(figsize=(8,8))
sns.set(style="darkgrid")
ax = sns.distplot(df_train['ecg'], label='Training dataset')
ax2 = sns.distplot(df_test['ecg'], label='Test dataset')
plt.legend()
plt.xlabel('ECG value')
plt.title('Distribution comparison of ECG of train and test dataset')
plt.show()
```



Observations:

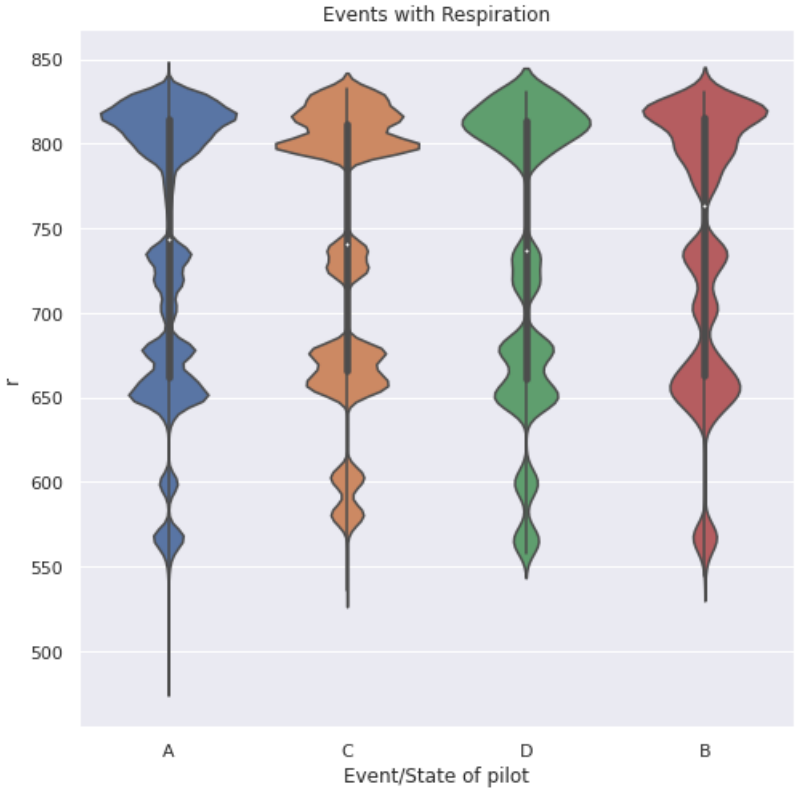
- For most of the part distributions are similar except after 18000 training data has more values.

Column - r (Respiration)

- The sensor had a resolution/bit of .2384186 μV and a range of -2.0V to +2.0V. The data are provided in microvolts.

In [ ]:

```
plt.figure(figsize=(8,8))
sns.set(style="darkgrid")
ax = sns.violinplot(x="event", y="r", data=df_train)
plt.xlabel('Event/State of pilot')
plt.title('Events with Respiration')
plt.show()
```

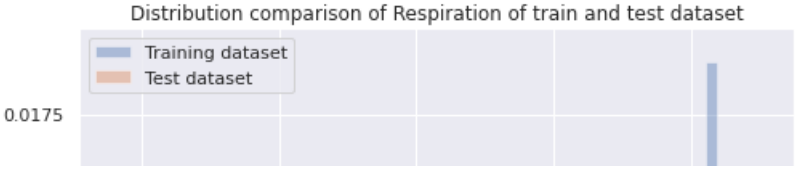


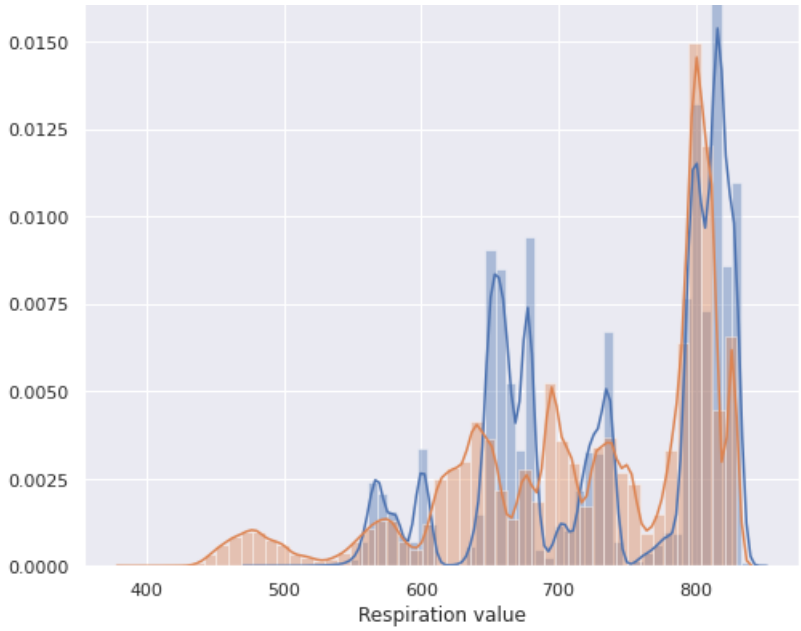
Observations:

- We can clearly see is the presence of some outliers in event A as the lower part of the range is reaching 450 while for other events it is very similar around 50
- For rest the data overlap is quite visible, even density distribution is quite similar.

In [ ]:

```
plt.figure(figsize=(8,8))
sns.set(style="darkgrid")
ax = sns.distplot(df_train['r'], label='Training dataset')
ax2 = sns.distplot(df_test['r'], label='Test dataset')
plt.legend()
plt.xlabel('Respiration value')
plt.title('Distribution comparison of Respiration of train and test dataset')
plt.show()
```





Observation:

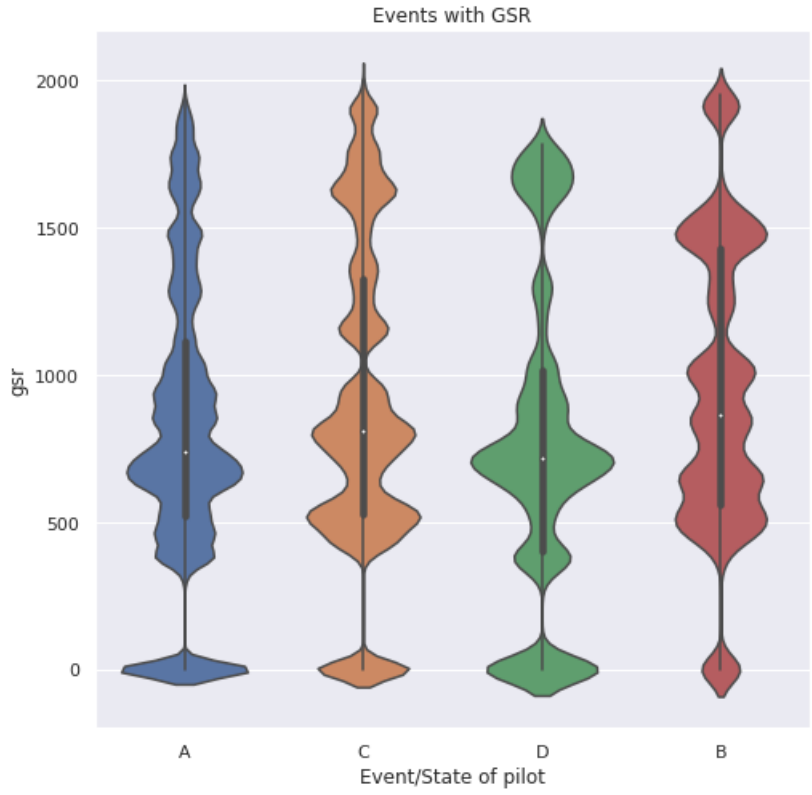
- Distribution of train and test data does not overlap. The density varies maximum in value below 550. This data below 550 can also not be considered as outliers as there are significant amount of values in that region.

Column-GSR

- Galvanic Skin Response, a measure of electrodermal activity. The sensor had a resolution/bit of .2384186 μV and a range of -2.0V to +2.0V. The data are provided in microvolts.

In [ ]:

```
plt.figure(figsize=(8,8))
sns.set(style="darkgrid")
ax = sns.violinplot(x="event", y="gsr", data=df_train)
plt.xlabel('Event/State of pilot')
plt.title('Events with GSR')
plt.show()
```

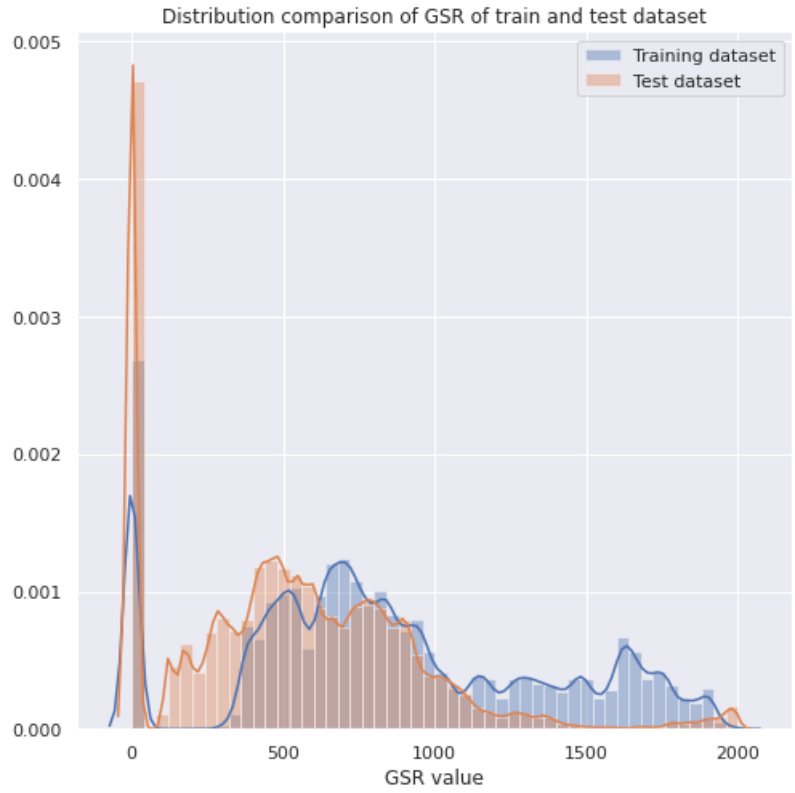


Observation:

- The overlapping of range is a little less as compared to other features.
- This could be a very useful feature to seperate out events in few range of values.
- Apart from that the distribution is quite similar.

In [ ]:

```
plt.figure(figsize=(8,8))
sns.set(style="darkgrid")
ax = sns.distplot(df_train['gsr'], label='Training dataset')
ax2 = sns.distplot(df_test['gsr'], label='Test dataset')
plt.legend()
plt.xlabel('GSR value')
plt.title('Distribution comparison of GSR of train and test dataset')
plt.show()
```



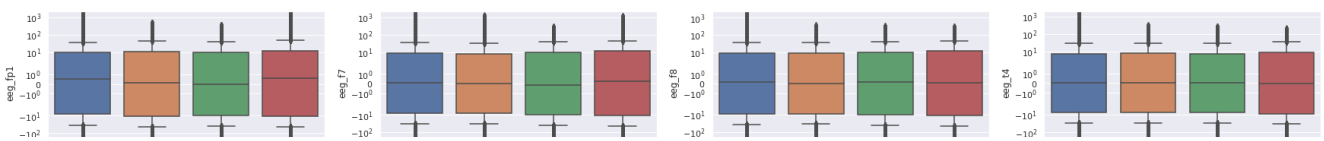
Observations:

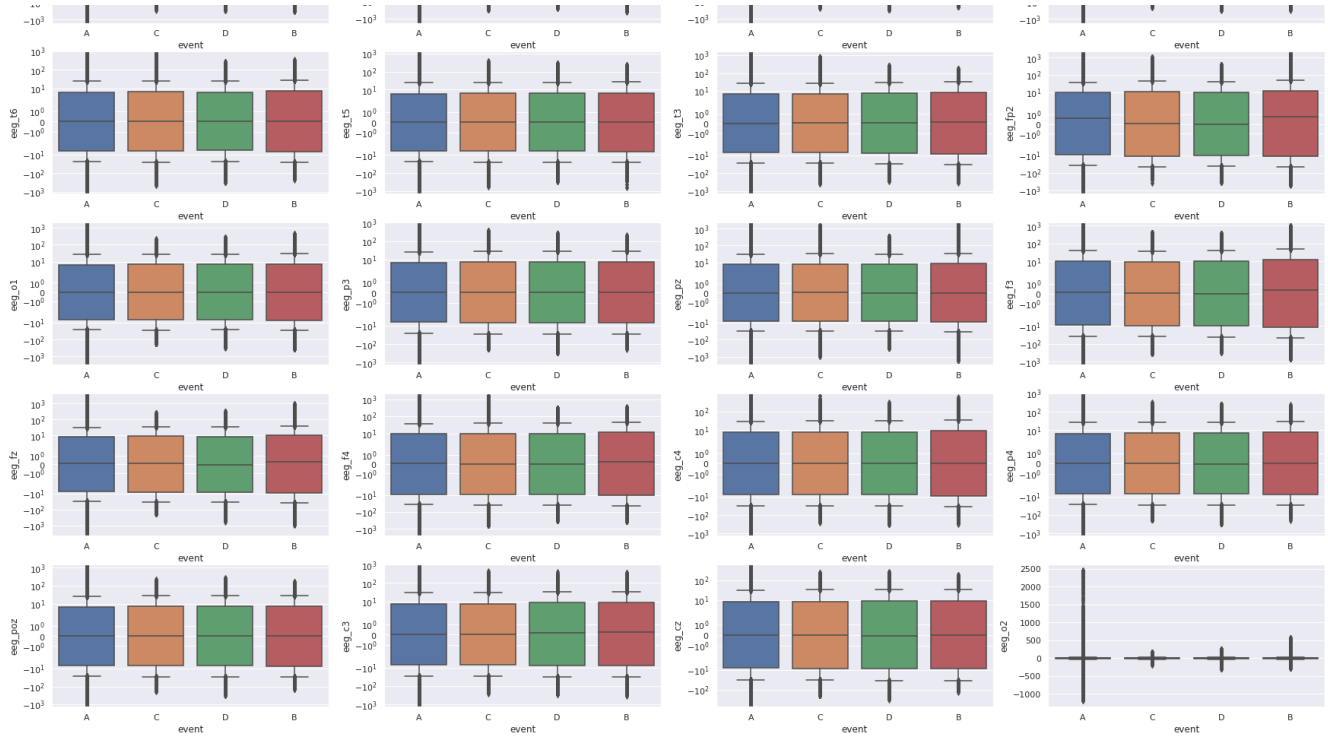
- The data distribution is not overlapping and it's quite different after the value of 1000 and below the value of 400 approximately.

Column - EEG (Electroencephalogram recordings)

In [ ]:

```
eeg_features = ["eeg_fp1","eeg_f7","eeg_f8","eeg_t4","eeg_t6","eeg_t5","eeg_t3","eeg_fp2","eeg_o1",
"eeg_p3","eeg_pz","eeg_f3","eeg_fz","eeg_f4","eeg_c4","eeg_p4","eeg_poz","eeg_c3","eeg_cz","eeg_o2"
]
plt.figure(figsize=(30,20))
for i,f in enumerate(eeg_features):
  plt.yscale('symlog')
  plt.subplot(5, 4, i+1)
  sns.boxplot(x="event", y=f, data=df_train)
plt.show()
```



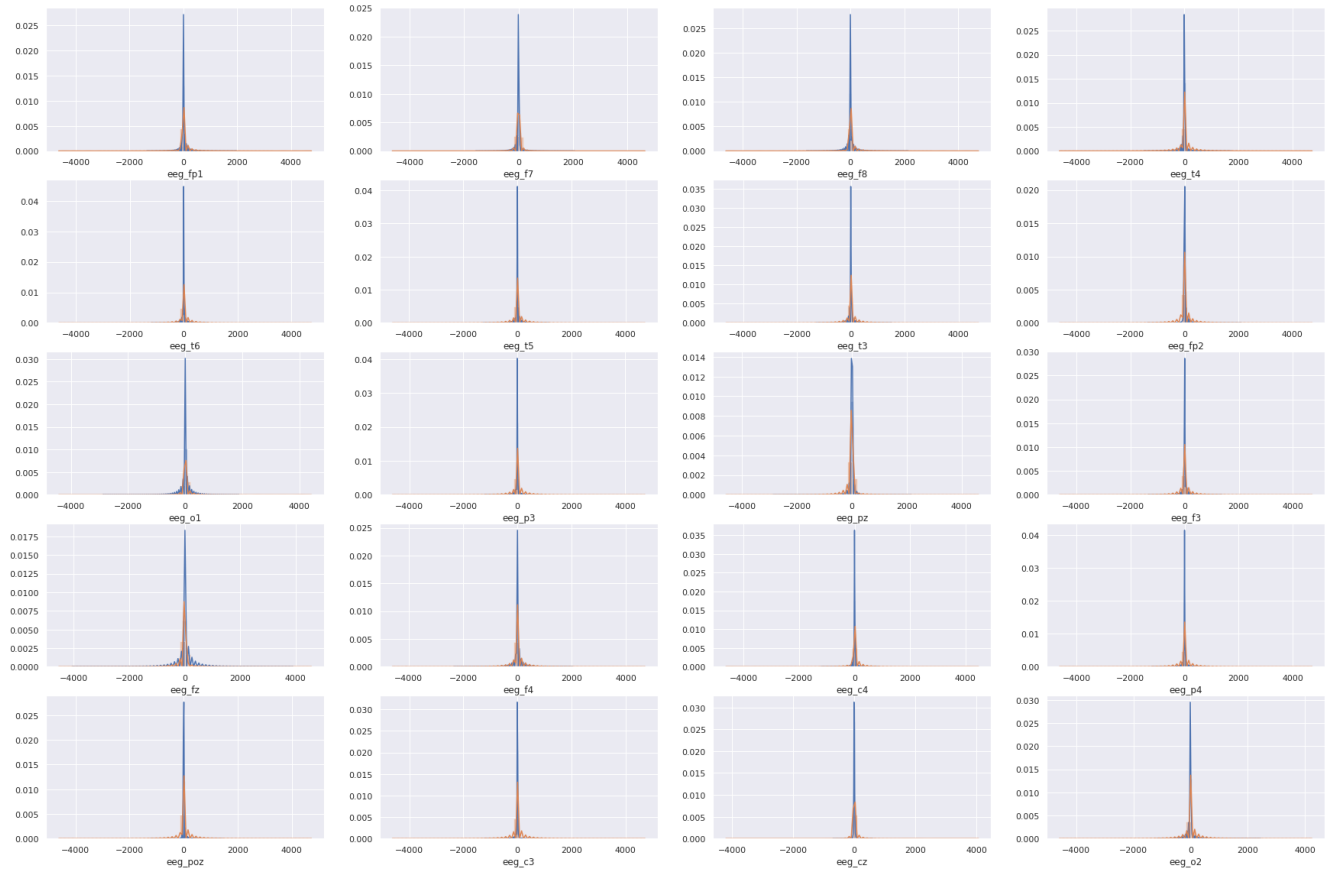


Observations:

- From almost all the eeg features we can see that they are mostly overlapping and the range for every event in every feature also varies to extreme ranges with some good density of points.

In [ ]:

```
plt.figure(figsize=(30,20))
for i,f in enumerate(eeg_features):
  plt.subplot(5, 4, i+1)
  sns.distplot(df_train[f])
  sns.distplot(df_test[f])
plt.show()
```



Observations:

- Mean value is almost 0 for all of them and the distribution is very similar in both train and test.

Checking Correlation among features

In [ ]:

```
corrMatrix = df_train.corr()
print(corrMatrix)
```

```
             crew      time      seat  ...       ecg         r       gsr
crew     1.000000  0.020509 -0.000026  ... -0.092310  0.017672  0.046665
time     0.020509  1.000000 -0.000092  ...  0.016148  0.002028 -0.023212
seat    -0.000026 -0.000092  1.000000  ...  0.065637  0.895856 -0.203039
eeg_fp1  0.004439 -0.001095  0.001293  ...  0.002471  0.001815 -0.005918
eeg_f7  -0.000304  0.000230  0.009259  ...  0.000509  0.006118  0.001325
eeg_f8   0.003582 -0.000951  0.004619  ... -0.002611  0.003651 -0.000659
eeg_t4  -0.000615 -0.001122  0.007370  ...  0.001917  0.005457 -0.002319
eeg_t6   0.009451 -0.000004  0.000428  ... -0.006497  0.000652 -0.002887
eeg_t5   0.004767  0.001654  0.005459  ... -0.001318  0.005259 -0.002809
eeg_t3  -0.001903  0.000063  0.007842  ...  0.003185  0.006459  0.000319
eeg_fp2  0.004089 -0.001906  0.002734  ...  0.001696  0.003869 -0.003926
eeg_o1   0.001793  0.001169  0.003672  ...  0.000896  0.001942 -0.000581
eeg_p3   0.002454  0.000479  0.002070  ... -0.003633  0.000845 -0.001831
eeg_pz   0.002228  0.000803  0.001865  ...  0.001788  0.002066 -0.000515
eeg_f3   0.003981  0.002119  0.002280  ... -0.003125  0.001427 -0.001427
eeg_fz   0.006493 -0.001744  0.000718  ... -0.000623 -0.000213  0.000853
eeg_f4   0.006101  0.000321  0.004490  ... -0.003315  0.004644 -0.000181
eeg_c4   0.011650 -0.000741 -0.002899  ... -0.011430 -0.003037 -0.006243
eeg_p4   0.006458  0.000037 -0.001174  ... -0.006457 -0.002102 -0.003424
eeg_poz  0.002011  0.000057  0.002245  ... -0.002110  0.000245 -0.001316
eeg_c3   0.006761  0.000538  0.001608  ... -0.006389  0.000883 -0.004067
eeg_cz   0.004878  0.000439  0.001167  ... -0.000852 -0.000049 -0.001177
eeg_o2   0.000739 -0.002336  0.005564  ...  0.001437  0.003492 -0.003213
ecg     -0.092310  0.016148  0.065637  ...  1.000000  0.049906  0.214831
r        0.017672  0.002028  0.895856  ...  0.049906  1.000000 -0.218775
gsr      0.046665 -0.023212 -0.203039  ...  0.214831 -0.218775  1.000000

[26 rows x 26 columns]
```

In [ ]:

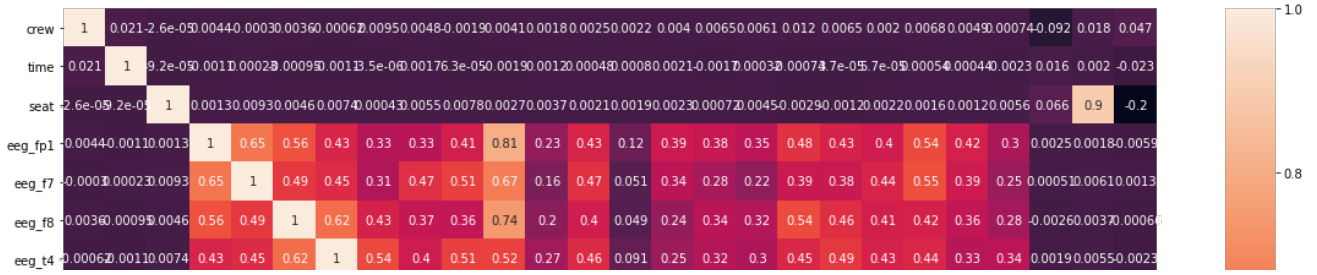
```
df_tr = df_train
df_tr.drop(columns=['crew'], inplace = True)
df_tr.drop(columns=['experiment'], inplace = True)
df_tr.drop(columns=['seat'], inplace = True)
```

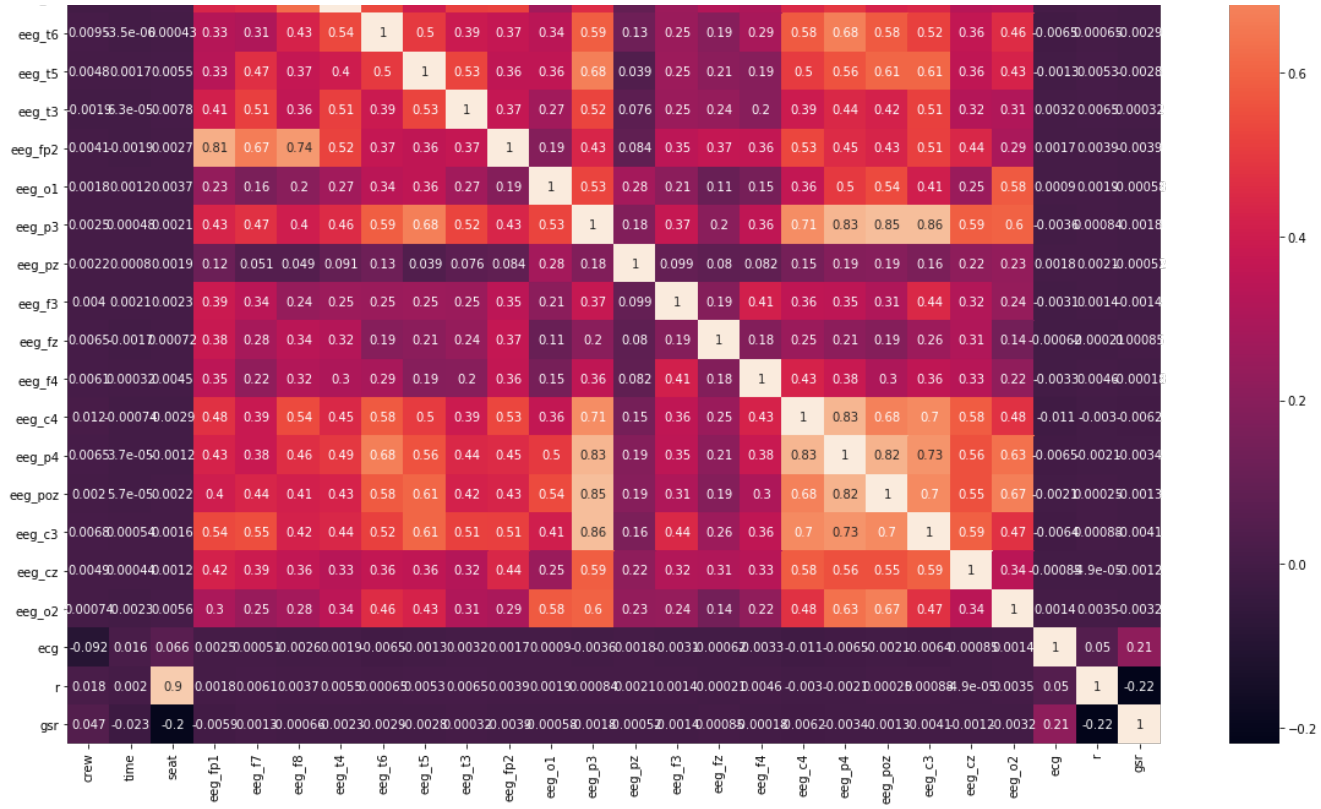
In [ ]:

```
labelencoder = LabelEncoder()
df_tr['event'] = labelencoder.fit_transform(df_tr['event'])
```

In [6]:

```
plt.figure(figsize=(20,15))
corrMatrix = df_train.corr()
sns.heatmap(corrMatrix, annot=True)
plt.show()
```





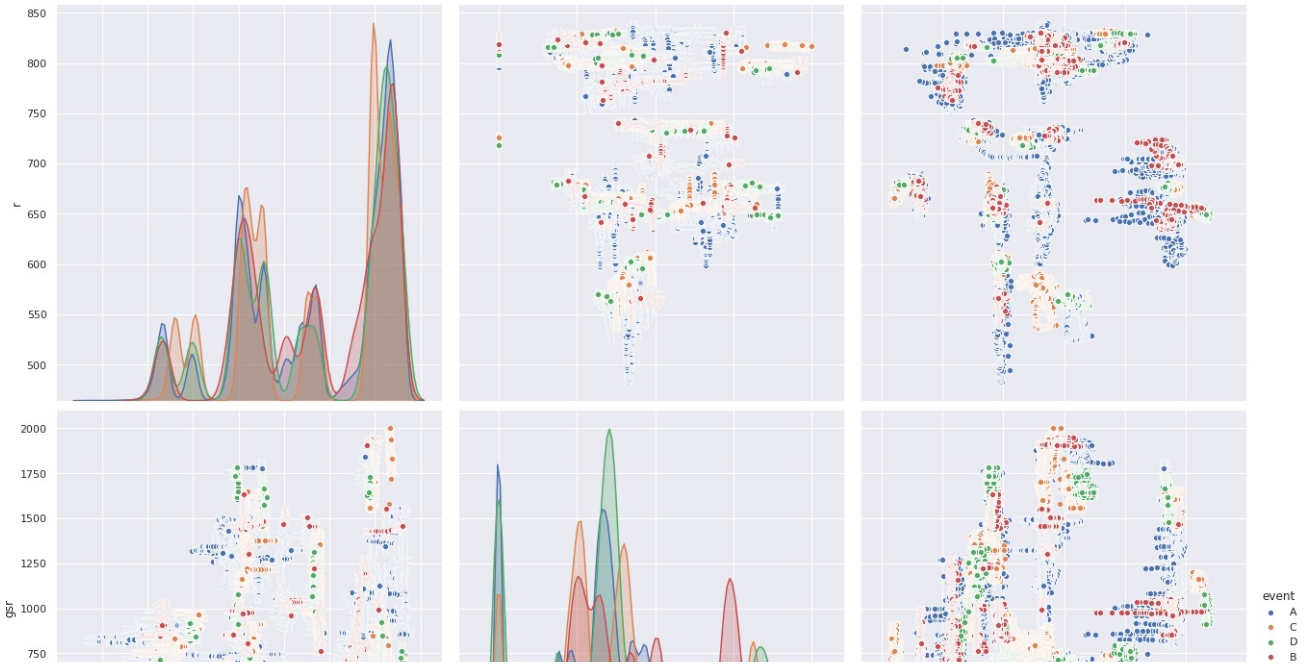
Observations:

- A positive correlation means that value of feature 1 increases as the value of feature 2 increases if they are highlt correlated.
- From above plot we cn see that there are few value i.e > 0.7 which are considered highly correlated features.
- In such case we can remove one of the correlated feature from the pair.
- Here seat and respiration are very highly corelated.
- Apart from that (ecg,gsr) and (respiration,gsr) alsohave a positive correlation but the value is low.
- For all the eeg features we can observe hogh correlation in many cases as it is understandable because they al are measured toghether in one setup and they just measure different aspects of electrical activity of brain.

Pair Plots

In [ ]:

```
sns.set(style="darkgrid")
sns.pairplot(df_tr,hue="event",size=6, vars=['r','gsr','ecg'])
plt.show()
```

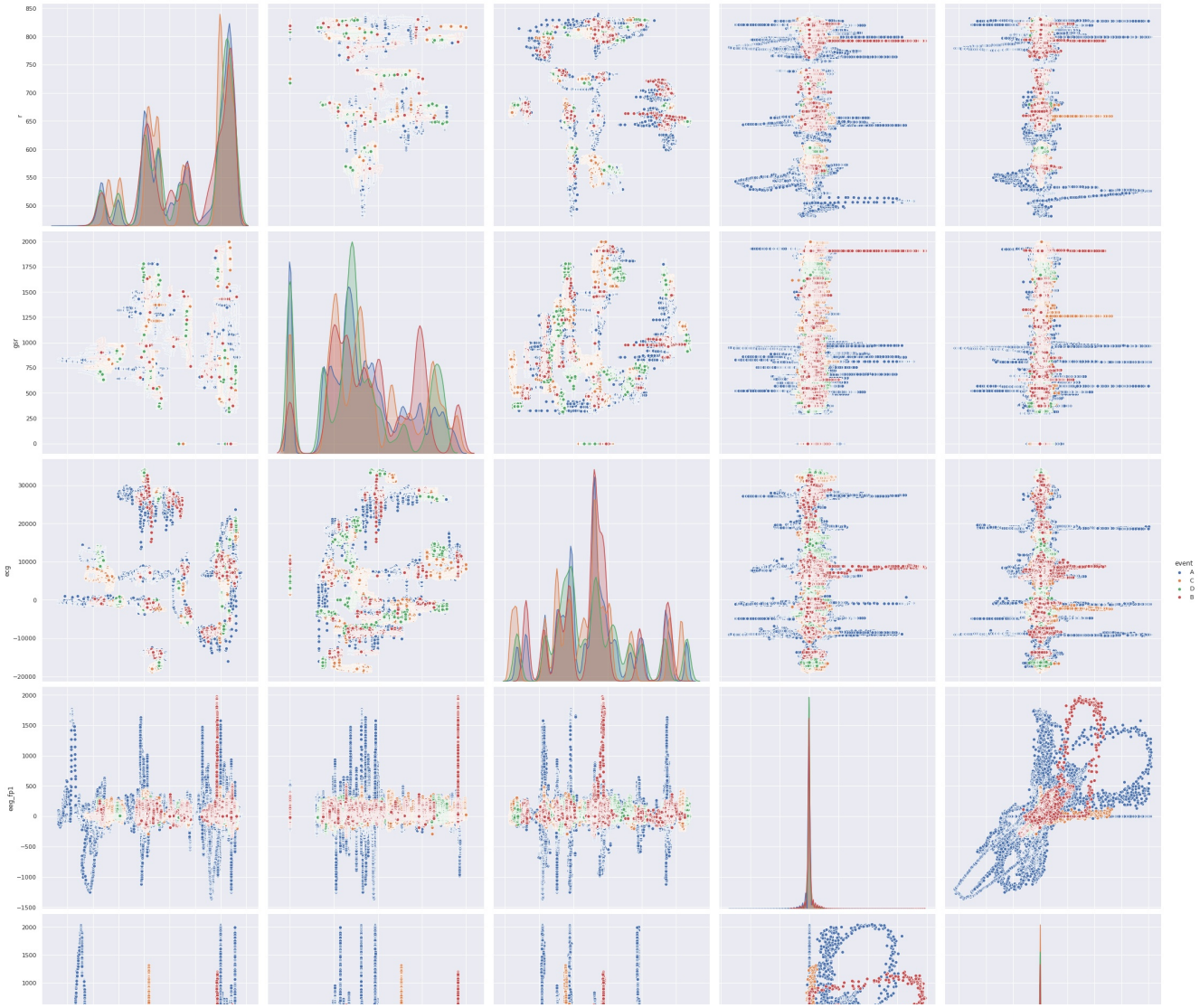


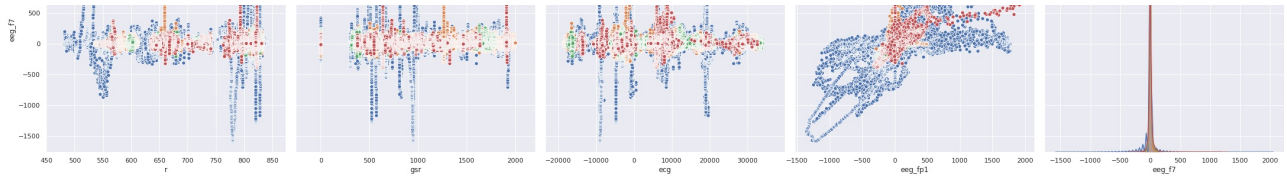
Observations:

- We can see in some pairplots like gsr and ecg some values are easily separable which can be useful.
- But for most of the part they are overlapping.

In [ ]:

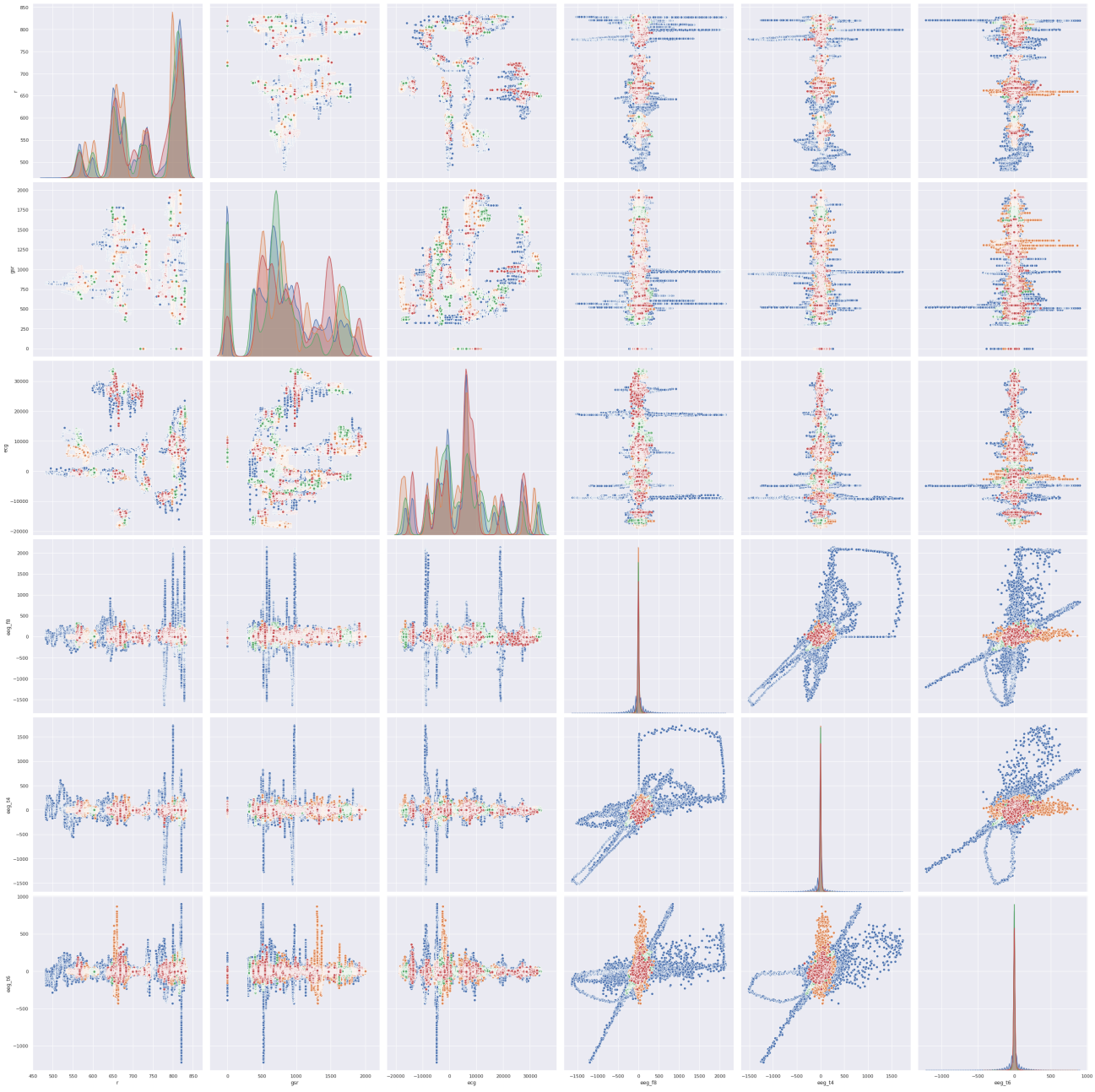
```
sns.set(style="darkgrid")
sns.pairplot(df_tr,hue="event",size=6, vars=['r','gsr','ecg','eeg_fp1','eeg_f7'])
plt.show()
```





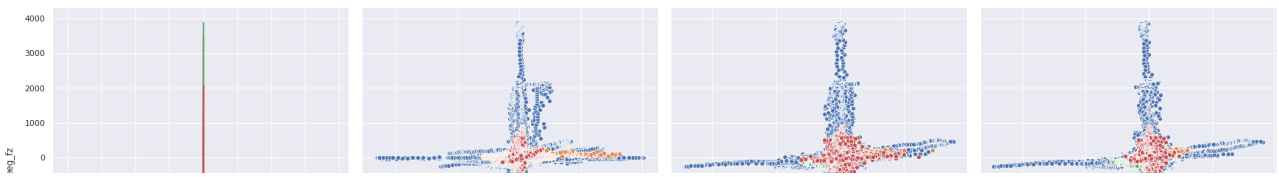
In [ ]:

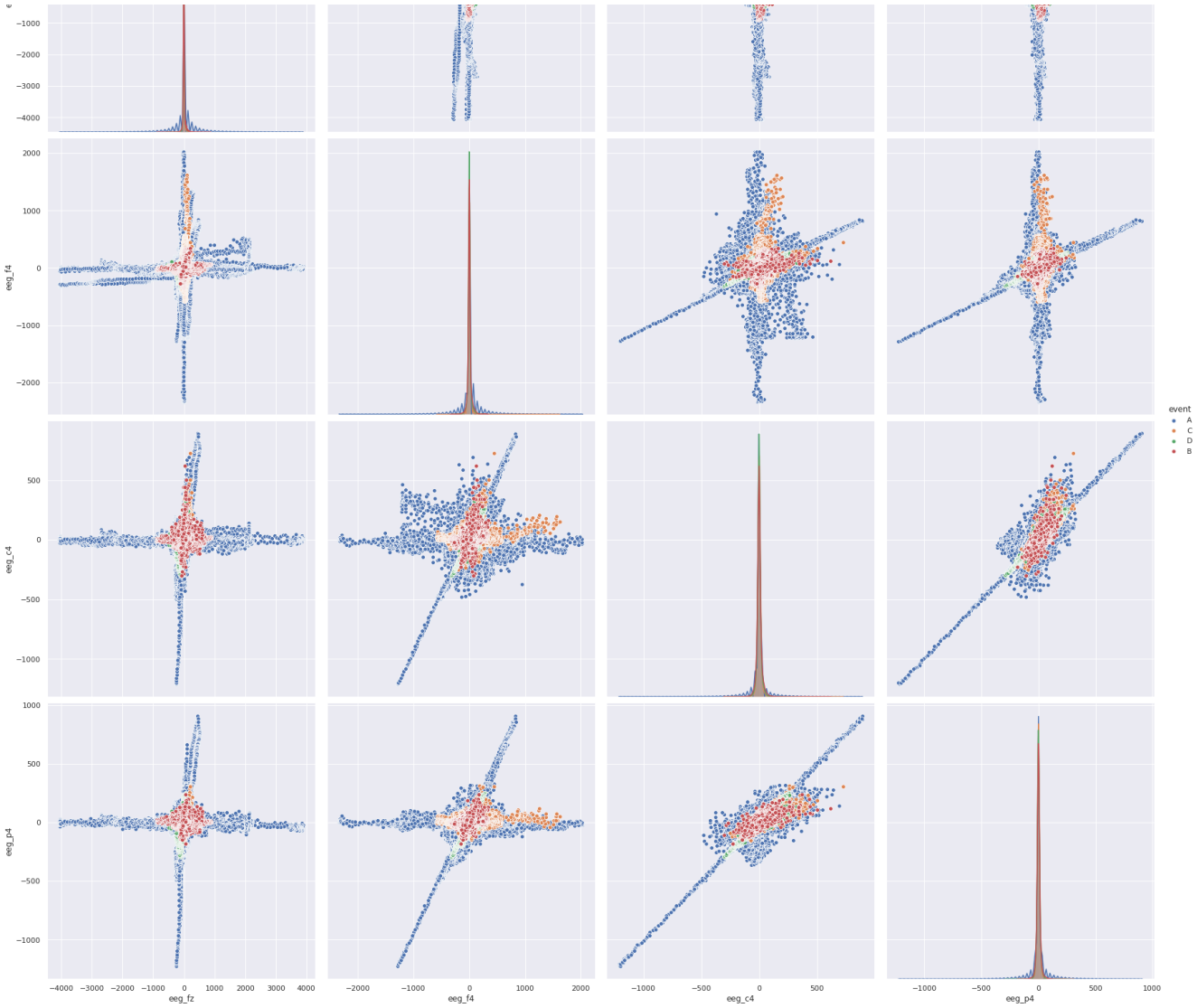
```
sns.set(style="darkgrid")
sns.pairplot(df_tr,hue="event",size=6, vars=['r','gsr','ecg','eeg_f8','eeg_t4','eeg_t6'])
plt.show()
```



In [ ]:

```
sns.set(style="darkgrid")
sns.pairplot(df_tr,hue="event",size=6, vars=['eeg_fz','eeg_f4','eeg_c4','eeg_p4'])
plt.show()
```





Observations:

- We can see a pattern in all the above graphs eeg can be useful in differentiating events when paired with other features.
- As baseline is the most common event so we can see it in most of the parts of any plot.
- But there are plots in which we can easily differentiate other events.

Phi_k Correlation matrix

- The calculation of correlation coefficients between paired data variables is a standard tool of analysis for every data analyst. Pearson's correlation coefficient is a de facto standard in most fields, but by construction only works for interval variables (sometimes called continuous variables). Pearson is unsuitable for data sets with mixed variable types, e.g. where some variables are ordinal or categorical.
- While many correlation coefficients exist, each with different features, we have not been able to find a correlation coefficient with Pearson-like characteristics and a sound statistical interpretation that works for interval, ordinal and categorical variable types alike.

In [39]:

```
df_train[:2000000].phik_matrix()
```

```
interval columns not set, guessing: ['crew', 'time', 'seat', 'eeg_fp1', 'eeg_f7', 'eeg_f8',
'eeg_t4', 'eeg_t6', 'eeg_t5', 'eeg_t3', 'eeg_fp2', 'eeg_o1', 'eeg_p3', 'eeg_pz', 'eeg_f3',
'eeg_fz', 'eeg_f4', 'eeg_c4', 'eeg_p4', 'eeg_poz', 'eeg_c3', 'eeg_cz', 'eeg_o2', 'ecg', 'r', 'gsr'
]
```

Out[39]:

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3
crew	1.000000	0.133278	0.102896	0.000000	0.080214	0.060921	0.093136	0.047568	0.078113	0.103175	0.054038

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3
experiment	0.133278	1.000000	0.120973	0.000000	0.101112	0.103024	0.108990	0.056685	0.089902	0.054128	0.065415
time	0.102896	0.120973	1.000000	0.000000	0.070412	0.063177	0.059252	0.053284	0.086290	0.071331	0.076940
seat	0.000000	0.000000	0.000000	1.000000	0.057822	0.077246	0.029811	0.033472	0.049302	0.065973	0.026156
eeg_fp1	0.080214	0.101112	0.070412	0.057822	1.000000	0.834519	0.793527	0.805384	0.810899	0.767909	0.726440
eeg_f7	0.060921	0.103024	0.063177	0.077246	0.834519	1.000000	0.874438	0.899064	0.893522	0.880618	0.834585
eeg_f8	0.093136	0.108990	0.059252	0.029811	0.793527	0.874438	1.000000	0.963090	0.975746	0.962850	0.926786
eeg_t4	0.047568	0.056685	0.053284	0.033472	0.805384	0.899064	0.963090	1.000000	0.962041	0.950172	0.920264
eeg_t6	0.078113	0.089902	0.086290	0.049302	0.810899	0.893522	0.975746	0.962041	1.000000	0.976358	0.932212
eeg_t5	0.103175	0.054128	0.071331	0.065973	0.767909	0.880618	0.962850	0.950172	0.976358	1.000000	0.923287
eeg_t3	0.054038	0.065415	0.076940	0.026156	0.726440	0.834585	0.926786	0.920264	0.932212	0.923287	1.000000
eeg_fp2	0.071597	0.089561	0.066097	0.040875	0.915116	0.866353	0.803792	0.831327	0.802811	0.765399	0.735230
eeg_o1	0.089974	0.079354	0.083177	0.049724	0.728682	0.799145	0.867080	0.849304	0.899526	0.866806	0.835988
eeg_p3	0.090009	0.059947	0.077626	0.060866	0.810880	0.900909	0.967450	0.963299	0.988523	0.975547	0.935057
eeg_pz	0.138412	0.055050	0.070395	0.099016	0.733912	0.445014	0.304619	0.326853	0.327386	0.299320	0.250440
eeg_f3	0.060552	0.084701	0.065232	0.078976	0.854239	0.931292	0.947750	0.957110	0.967419	0.955850	0.926431
eeg_fz	0.194617	0.082247	0.085210	0.154006	0.681678	0.609686	0.691954	0.630630	0.652069	0.630687	0.535670
eeg_f4	0.072989	0.080420	0.065856	0.055005	0.534549	0.636066	0.695816	0.691639	0.708941	0.689757	0.664948
eeg_c4	0.076506	0.095195	0.068356	0.072748	0.807781	0.896963	0.973434	0.963571	0.990033	0.975692	0.932710
eeg_p4	0.086758	0.074046	0.080893	0.064867	0.818913	0.906390	0.975434	0.964442	0.995200	0.979972	0.934367
eeg_poz	0.077103	0.059509	0.078531	0.046253	0.821953	0.882469	0.921043	0.915600	0.932630	0.912781	0.906396
eeg_c3	0.076900	0.070963	0.071890	0.056034	0.817271	0.903714	0.966065	0.958183	0.988619	0.974668	0.942875
eeg_cz	0.225179	0.061642	0.079392	0.170803	0.591620	0.715535	0.493065	0.570047	0.542464	0.472724	0.432300
eeg_o2	0.076770	0.065282	0.076813	0.048689	0.727563	0.741308	0.783419	0.771767	0.795981	0.821995	0.744580
ecg	0.830050	0.595114	0.224515	0.769679	0.084467	0.072866	0.098622	0.064333	0.103666	0.126611	0.062616
r	0.796347	0.349892	0.188082	1.000000	0.419095	0.217437	0.091187	0.147479	0.099103	0.100442	0.132306
gsr	0.756444	0.474171	0.382382	0.980133	0.110833	0.094609	0.104231	0.052131	0.094822	0.120515	0.084325
event	0.159654	0.701888	0.272999	0.000000	0.043305	0.064734	0.055521	0.027927	0.047095	0.028961	0.026368

Observations:

- Phi_k correlation matrix gives us better understanding and more accurate values for correlation because we have a combination of numerical and categorical features.
- Here we can observe that there are many pair of features with very high as well as very low correlation.

PCA for feature importance

In [ ]:

```
df_tr.head(0)
```

Out[ ]:

	time	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2	eeg_o1	eeg_p3	eeg_pz	eeg_f3	eeg_fz	eeg_f4	ec

In [ ]:

```
x = StandardScaler().fit_transform(df_tr)
pca = PCA(n_components=3)
x_new = pca.fit_transform(x)
```

In [ ]:

In [ ]:

```
print('Explained variation per principal component: {}'.format(pca.explained_variance_ratio_))
```

```
Explained variation per principal component: [0.35720773 0.07866801 0.05134712]
```

In [ ]:

```
np.set_printoptions(suppress=True)
print(abs(pca.components_))
```

```
[[0.00007607 0.22763804 0.21790449 0.2207211  0.22110877 0.23040171
  0.22981337 0.20674459 0.23711182 0.17456232 0.29153342 0.07251909
  0.16317891 0.12859652 0.15893993 0.27274693 0.28803203 0.27737586
  0.28302925 0.22200082 0.21304608 0.00119598 0.00112271 0.00132099
  0.00209688]
 [0.00229749 0.34710986 0.30143614 0.30711672 0.16782412 0.12160947
  0.10708022 0.06635688 0.38305463 0.332007   0.21787631 0.17412993
  0.09618877 0.26934356 0.10221629 0.06967639 0.21168404 0.24608663
  0.07022998 0.00444165 0.30478126 0.00571034 0.00520795 0.00043838
  0.01943044]
 [0.02989184 0.00729432 0.00210959 0.00309358 0.01269933 0.00230873
  0.00353531 0.00741346 0.00529362 0.00340658 0.0000238  0.00400973
  0.00708744 0.01295895 0.00948478 0.00415051 0.00034643 0.0000031
  0.00245473 0.00411964 0.00278727 0.45776235 0.48096578 0.74518505
  0.04709602]]
```

Observations:

- The importance of each feature is reflected by the magnitude of the corresponding values in the eigenvectors (higher magnitude - higher importance).
- PC1 explains 35% of the data.
- pca.components_ has shape [n_components, n_features]
- Look at the absolute values of the Eigenvectors components corresponding to the k largest Eigenvalues
- So for each PC the higher the value the more important is the feature.

TSNE for visualization of data

In [ ]:

```
labelencoder = LabelEncoder()
df_tr_tsne = df_train.sample(frac=0.05, random_state=1)
df_tr_tsne['event'] = labelencoder.fit_transform(df_tr_tsne['event'])
df_tr_tsne['experiment'] = labelencoder.fit_transform(df_tr_tsne['experiment'])
```

In [ ]:

```
tsne = TSNE(n_components=2, perplexity=40, n_iter=250,verbose=1)
tsne_results = tsne.fit_transform(df_tr_tsne)
```

```
[t-SNE] Computing 121 nearest neighbors...
[t-SNE] Indexed 243371 samples in 3.956s...
[t-SNE] Computed neighbors for 243371 samples in 150.077s...
[t-SNE] Computed conditional probabilities for sample 1000 / 243371
[t-SNE] Computed conditional probabilities for sample 2000 / 243371
[t-SNE] Computed conditional probabilities for sample 3000 / 243371
[t-SNE] Computed conditional probabilities for sample 4000 / 243371
[t-SNE] Computed conditional probabilities for sample 5000 / 243371
[t-SNE] Computed conditional probabilities for sample 6000 / 243371
[t-SNE] Computed conditional probabilities for sample 7000 / 243371
[t-SNE] Computed conditional probabilities for sample 8000 / 243371
[t-SNE] Computed conditional probabilities for sample 9000 / 243371
[t-SNE] Computed conditional probabilities for sample 10000 / 243371
[t-SNE] Computed conditional probabilities for sample 11000 / 243371
[t-SNE] Computed conditional probabilities for sample 12000 / 243371
[t-SNE] Computed conditional probabilities for sample 13000 / 243371
[t-SNE] Computed conditional probabilities for sample 14000 / 243371
[t-SNE] Computed conditional probabilities for sample 15000 / 243371
[t-SNE] Computed conditional probabilities for sample 16000 / 243371
[t-SNE] Computed conditional probabilities for sample 17000 / 243371
[t-SNE] Computed conditional probabilities for sample 18000 / 243371
```

```
[t-SNE] Computed conditional probabilities for sample 19000 / 243371
[t-SNE] Computed conditional probabilities for sample 20000 / 243371
[t-SNE] Computed conditional probabilities for sample 21000 / 243371
[t-SNE] Computed conditional probabilities for sample 22000 / 243371
[t-SNE] Computed conditional probabilities for sample 23000 / 243371
[t-SNE] Computed conditional probabilities for sample 24000 / 243371
[t-SNE] Computed conditional probabilities for sample 25000 / 243371
[t-SNE] Computed conditional probabilities for sample 26000 / 243371
[t-SNE] Computed conditional probabilities for sample 27000 / 243371
[t-SNE] Computed conditional probabilities for sample 28000 / 243371
[t-SNE] Computed conditional probabilities for sample 29000 / 243371
[t-SNE] Computed conditional probabilities for sample 30000 / 243371
[t-SNE] Computed conditional probabilities for sample 31000 / 243371
[t-SNE] Computed conditional probabilities for sample 32000 / 243371
[t-SNE] Computed conditional probabilities for sample 33000 / 243371
[t-SNE] Computed conditional probabilities for sample 34000 / 243371
[t-SNE] Computed conditional probabilities for sample 35000 / 243371
[t-SNE] Computed conditional probabilities for sample 36000 / 243371
[t-SNE] Computed conditional probabilities for sample 37000 / 243371
[t-SNE] Computed conditional probabilities for sample 38000 / 243371
[t-SNE] Computed conditional probabilities for sample 39000 / 243371
[t-SNE] Computed conditional probabilities for sample 40000 / 243371
[t-SNE] Computed conditional probabilities for sample 41000 / 243371
[t-SNE] Computed conditional probabilities for sample 42000 / 243371
[t-SNE] Computed conditional probabilities for sample 43000 / 243371
[t-SNE] Computed conditional probabilities for sample 44000 / 243371
[t-SNE] Computed conditional probabilities for sample 45000 / 243371
[t-SNE] Computed conditional probabilities for sample 46000 / 243371
[t-SNE] Computed conditional probabilities for sample 47000 / 243371
[t-SNE] Computed conditional probabilities for sample 48000 / 243371
[t-SNE] Computed conditional probabilities for sample 49000 / 243371
[t-SNE] Computed conditional probabilities for sample 50000 / 243371
[t-SNE] Computed conditional probabilities for sample 51000 / 243371
[t-SNE] Computed conditional probabilities for sample 52000 / 243371
[t-SNE] Computed conditional probabilities for sample 53000 / 243371
[t-SNE] Computed conditional probabilities for sample 54000 / 243371
[t-SNE] Computed conditional probabilities for sample 55000 / 243371
[t-SNE] Computed conditional probabilities for sample 56000 / 243371
[t-SNE] Computed conditional probabilities for sample 57000 / 243371
[t-SNE] Computed conditional probabilities for sample 58000 / 243371
[t-SNE] Computed conditional probabilities for sample 59000 / 243371
[t-SNE] Computed conditional probabilities for sample 60000 / 243371
[t-SNE] Computed conditional probabilities for sample 61000 / 243371
[t-SNE] Computed conditional probabilities for sample 62000 / 243371
[t-SNE] Computed conditional probabilities for sample 63000 / 243371
[t-SNE] Computed conditional probabilities for sample 64000 / 243371
[t-SNE] Computed conditional probabilities for sample 65000 / 243371
[t-SNE] Computed conditional probabilities for sample 66000 / 243371
[t-SNE] Computed conditional probabilities for sample 67000 / 243371
[t-SNE] Computed conditional probabilities for sample 68000 / 243371
[t-SNE] Computed conditional probabilities for sample 69000 / 243371
[t-SNE] Computed conditional probabilities for sample 70000 / 243371
[t-SNE] Computed conditional probabilities for sample 71000 / 243371
[t-SNE] Computed conditional probabilities for sample 72000 / 243371
[t-SNE] Computed conditional probabilities for sample 73000 / 243371
[t-SNE] Computed conditional probabilities for sample 74000 / 243371
[t-SNE] Computed conditional probabilities for sample 75000 / 243371
[t-SNE] Computed conditional probabilities for sample 76000 / 243371
[t-SNE] Computed conditional probabilities for sample 77000 / 243371
[t-SNE] Computed conditional probabilities for sample 78000 / 243371
[t-SNE] Computed conditional probabilities for sample 79000 / 243371
[t-SNE] Computed conditional probabilities for sample 80000 / 243371
[t-SNE] Computed conditional probabilities for sample 81000 / 243371
[t-SNE] Computed conditional probabilities for sample 82000 / 243371
[t-SNE] Computed conditional probabilities for sample 83000 / 243371
[t-SNE] Computed conditional probabilities for sample 84000 / 243371
[t-SNE] Computed conditional probabilities for sample 85000 / 243371
[t-SNE] Computed conditional probabilities for sample 86000 / 243371
[t-SNE] Computed conditional probabilities for sample 87000 / 243371
[t-SNE] Computed conditional probabilities for sample 88000 / 243371
[t-SNE] Computed conditional probabilities for sample 89000 / 243371
[t-SNE] Computed conditional probabilities for sample 90000 / 243371
[t-SNE] Computed conditional probabilities for sample 91000 / 243371
[t-SNE] Computed conditional probabilities for sample 92000 / 243371
[t-SNE] Computed conditional probabilities for sample 93000 / 243371
[t-SNE] Computed conditional probabilities for sample 94000 / 243371
[t-SNE] Computed conditional probabilities for sample 95000 / 243371
```

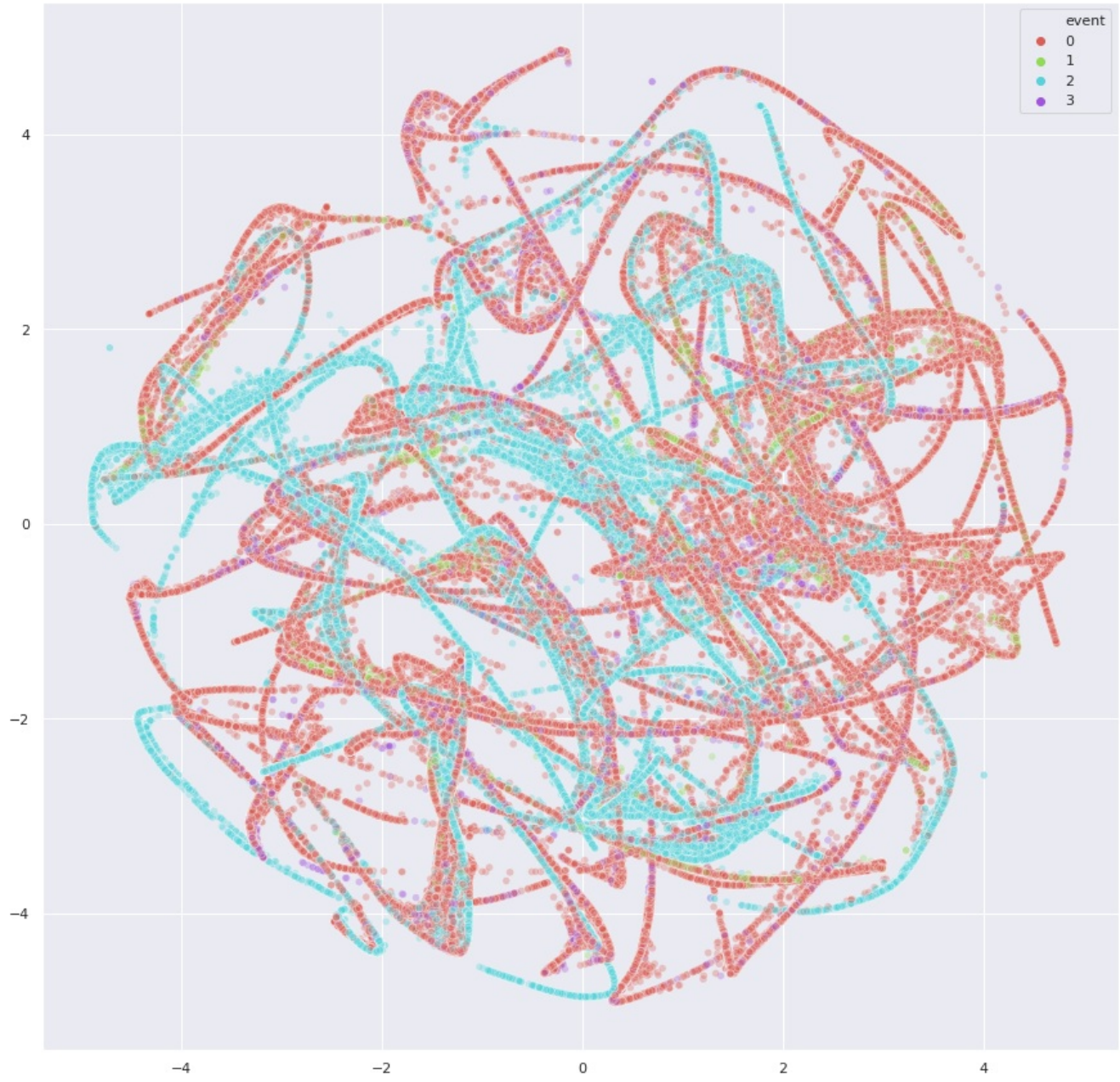
```
[t-SNE] Computed conditional probabilities for sample 96000 / 243371
[t-SNE] Computed conditional probabilities for sample 97000 / 243371
[t-SNE] Computed conditional probabilities for sample 98000 / 243371
[t-SNE] Computed conditional probabilities for sample 99000 / 243371
[t-SNE] Computed conditional probabilities for sample 100000 / 243371
[t-SNE] Computed conditional probabilities for sample 101000 / 243371
[t-SNE] Computed conditional probabilities for sample 102000 / 243371
[t-SNE] Computed conditional probabilities for sample 103000 / 243371
[t-SNE] Computed conditional probabilities for sample 104000 / 243371
[t-SNE] Computed conditional probabilities for sample 105000 / 243371
[t-SNE] Computed conditional probabilities for sample 106000 / 243371
[t-SNE] Computed conditional probabilities for sample 107000 / 243371
[t-SNE] Computed conditional probabilities for sample 108000 / 243371
[t-SNE] Computed conditional probabilities for sample 109000 / 243371
[t-SNE] Computed conditional probabilities for sample 110000 / 243371
[t-SNE] Computed conditional probabilities for sample 111000 / 243371
[t-SNE] Computed conditional probabilities for sample 112000 / 243371
[t-SNE] Computed conditional probabilities for sample 113000 / 243371
[t-SNE] Computed conditional probabilities for sample 114000 / 243371
[t-SNE] Computed conditional probabilities for sample 115000 / 243371
[t-SNE] Computed conditional probabilities for sample 116000 / 243371
[t-SNE] Computed conditional probabilities for sample 117000 / 243371
[t-SNE] Computed conditional probabilities for sample 118000 / 243371
[t-SNE] Computed conditional probabilities for sample 119000 / 243371
[t-SNE] Computed conditional probabilities for sample 120000 / 243371
[t-SNE] Computed conditional probabilities for sample 121000 / 243371
[t-SNE] Computed conditional probabilities for sample 122000 / 243371
[t-SNE] Computed conditional probabilities for sample 123000 / 243371
[t-SNE] Computed conditional probabilities for sample 124000 / 243371
[t-SNE] Computed conditional probabilities for sample 125000 / 243371
[t-SNE] Computed conditional probabilities for sample 126000 / 243371
[t-SNE] Computed conditional probabilities for sample 127000 / 243371
[t-SNE] Computed conditional probabilities for sample 128000 / 243371
[t-SNE] Computed conditional probabilities for sample 129000 / 243371
[t-SNE] Computed conditional probabilities for sample 130000 / 243371
[t-SNE] Computed conditional probabilities for sample 131000 / 243371
[t-SNE] Computed conditional probabilities for sample 132000 / 243371
[t-SNE] Computed conditional probabilities for sample 133000 / 243371
[t-SNE] Computed conditional probabilities for sample 134000 / 243371
[t-SNE] Computed conditional probabilities for sample 135000 / 243371
[t-SNE] Computed conditional probabilities for sample 136000 / 243371
[t-SNE] Computed conditional probabilities for sample 137000 / 243371
[t-SNE] Computed conditional probabilities for sample 138000 / 243371
[t-SNE] Computed conditional probabilities for sample 139000 / 243371
[t-SNE] Computed conditional probabilities for sample 140000 / 243371
[t-SNE] Computed conditional probabilities for sample 141000 / 243371
[t-SNE] Computed conditional probabilities for sample 142000 / 243371
[t-SNE] Computed conditional probabilities for sample 143000 / 243371
[t-SNE] Computed conditional probabilities for sample 144000 / 243371
[t-SNE] Computed conditional probabilities for sample 145000 / 243371
[t-SNE] Computed conditional probabilities for sample 146000 / 243371
[t-SNE] Computed conditional probabilities for sample 147000 / 243371
[t-SNE] Computed conditional probabilities for sample 148000 / 243371
[t-SNE] Computed conditional probabilities for sample 149000 / 243371
[t-SNE] Computed conditional probabilities for sample 150000 / 243371
[t-SNE] Computed conditional probabilities for sample 151000 / 243371
[t-SNE] Computed conditional probabilities for sample 152000 / 243371
[t-SNE] Computed conditional probabilities for sample 153000 / 243371
[t-SNE] Computed conditional probabilities for sample 154000 / 243371
[t-SNE] Computed conditional probabilities for sample 155000 / 243371
[t-SNE] Computed conditional probabilities for sample 156000 / 243371
[t-SNE] Computed conditional probabilities for sample 157000 / 243371
[t-SNE] Computed conditional probabilities for sample 158000 / 243371
[t-SNE] Computed conditional probabilities for sample 159000 / 243371
[t-SNE] Computed conditional probabilities for sample 160000 / 243371
[t-SNE] Computed conditional probabilities for sample 161000 / 243371
[t-SNE] Computed conditional probabilities for sample 162000 / 243371
[t-SNE] Computed conditional probabilities for sample 163000 / 243371
[t-SNE] Computed conditional probabilities for sample 164000 / 243371
[t-SNE] Computed conditional probabilities for sample 165000 / 243371
[t-SNE] Computed conditional probabilities for sample 166000 / 243371
[t-SNE] Computed conditional probabilities for sample 167000 / 243371
[t-SNE] Computed conditional probabilities for sample 168000 / 243371
[t-SNE] Computed conditional probabilities for sample 169000 / 243371
[t-SNE] Computed conditional probabilities for sample 170000 / 243371
[t-SNE] Computed conditional probabilities for sample 171000 / 243371
[t-SNE] Computed conditional probabilities for sample 172000 / 243371
```

[t-SNE] Computed conditional probabilities for sample 173000 / 243371
[t-SNE] Computed conditional probabilities for sample 174000 / 243371
[t-SNE] Computed conditional probabilities for sample 175000 / 243371
[t-SNE] Computed conditional probabilities for sample 176000 / 243371
[t-SNE] Computed conditional probabilities for sample 177000 / 243371
[t-SNE] Computed conditional probabilities for sample 178000 / 243371
[t-SNE] Computed conditional probabilities for sample 179000 / 243371
[t-SNE] Computed conditional probabilities for sample 180000 / 243371
[t-SNE] Computed conditional probabilities for sample 181000 / 243371
[t-SNE] Computed conditional probabilities for sample 182000 / 243371
[t-SNE] Computed conditional probabilities for sample 183000 / 243371
[t-SNE] Computed conditional probabilities for sample 184000 / 243371
[t-SNE] Computed conditional probabilities for sample 185000 / 243371
[t-SNE] Computed conditional probabilities for sample 186000 / 243371
[t-SNE] Computed conditional probabilities for sample 187000 / 243371
[t-SNE] Computed conditional probabilities for sample 188000 / 243371
[t-SNE] Computed conditional probabilities for sample 189000 / 243371
[t-SNE] Computed conditional probabilities for sample 190000 / 243371
[t-SNE] Computed conditional probabilities for sample 191000 / 243371
[t-SNE] Computed conditional probabilities for sample 192000 / 243371
[t-SNE] Computed conditional probabilities for sample 193000 / 243371
[t-SNE] Computed conditional probabilities for sample 194000 / 243371
[t-SNE] Computed conditional probabilities for sample 195000 / 243371
[t-SNE] Computed conditional probabilities for sample 196000 / 243371
[t-SNE] Computed conditional probabilities for sample 197000 / 243371
[t-SNE] Computed conditional probabilities for sample 198000 / 243371
[t-SNE] Computed conditional probabilities for sample 199000 / 243371
[t-SNE] Computed conditional probabilities for sample 200000 / 243371
[t-SNE] Computed conditional probabilities for sample 201000 / 243371
[t-SNE] Computed conditional probabilities for sample 202000 / 243371
[t-SNE] Computed conditional probabilities for sample 203000 / 243371
[t-SNE] Computed conditional probabilities for sample 204000 / 243371
[t-SNE] Computed conditional probabilities for sample 205000 / 243371
[t-SNE] Computed conditional probabilities for sample 206000 / 243371
[t-SNE] Computed conditional probabilities for sample 207000 / 243371
[t-SNE] Computed conditional probabilities for sample 208000 / 243371
[t-SNE] Computed conditional probabilities for sample 209000 / 243371
[t-SNE] Computed conditional probabilities for sample 210000 / 243371
[t-SNE] Computed conditional probabilities for sample 211000 / 243371
[t-SNE] Computed conditional probabilities for sample 212000 / 243371
[t-SNE] Computed conditional probabilities for sample 213000 / 243371
[t-SNE] Computed conditional probabilities for sample 214000 / 243371
[t-SNE] Computed conditional probabilities for sample 215000 / 243371
[t-SNE] Computed conditional probabilities for sample 216000 / 243371
[t-SNE] Computed conditional probabilities for sample 217000 / 243371
[t-SNE] Computed conditional probabilities for sample 218000 / 243371
[t-SNE] Computed conditional probabilities for sample 219000 / 243371
[t-SNE] Computed conditional probabilities for sample 220000 / 243371
[t-SNE] Computed conditional probabilities for sample 221000 / 243371
[t-SNE] Computed conditional probabilities for sample 222000 / 243371
[t-SNE] Computed conditional probabilities for sample 223000 / 243371
[t-SNE] Computed conditional probabilities for sample 224000 / 243371
[t-SNE] Computed conditional probabilities for sample 225000 / 243371
[t-SNE] Computed conditional probabilities for sample 226000 / 243371
[t-SNE] Computed conditional probabilities for sample 227000 / 243371
[t-SNE] Computed conditional probabilities for sample 228000 / 243371
[t-SNE] Computed conditional probabilities for sample 229000 / 243371
[t-SNE] Computed conditional probabilities for sample 230000 / 243371
[t-SNE] Computed conditional probabilities for sample 231000 / 243371
[t-SNE] Computed conditional probabilities for sample 232000 / 243371
[t-SNE] Computed conditional probabilities for sample 233000 / 243371
[t-SNE] Computed conditional probabilities for sample 234000 / 243371
[t-SNE] Computed conditional probabilities for sample 235000 / 243371
[t-SNE] Computed conditional probabilities for sample 236000 / 243371
[t-SNE] Computed conditional probabilities for sample 237000 / 243371
[t-SNE] Computed conditional probabilities for sample 238000 / 243371
[t-SNE] Computed conditional probabilities for sample 239000 / 243371
[t-SNE] Computed conditional probabilities for sample 240000 / 243371
[t-SNE] Computed conditional probabilities for sample 241000 / 243371
[t-SNE] Computed conditional probabilities for sample 242000 / 243371
[t-SNE] Computed conditional probabilities for sample 243000 / 243371
[t-SNE] Computed conditional probabilities for sample 243371 / 243371
[t-SNE] Mean sigma: 19.920914
[t-SNE] KL divergence after 250 iterations with early exaggeration: 104.510269
[t-SNE] KL divergence after 251 iterations:
17976931348623157081452742373170435679807056752584499659891747680315726078002853876058955863276687845895351438246423432132688946418276846754670353751698604991057655128207624549009038932894407586855

```
513394230458323690322294816580855933212334827479782620414472316873817718091929988125040402618412485
.000000
```

In [ ]:

```
sns.set(style="darkgrid")
plt.figure(figsize=(15,15))
sns.scatterplot(x=tsne_results[:,0], y=tsne_results[:,1],hue="event",palette=sns.color_palette("hls
", 4),data=df_tr_tsne,legend="full",alpha=0.3)
plt.show()
```



Observations:

- Due to memory and computing constraints we can only visualize a part of the data.
- From this we can easily see the dominant baseline state.
- Rest we cannot say anything conclusively as this is only projection of a little part of data.

Time series data analysis of features

Encoding categorical features for better understanding

In [7]:

```
df_tr_ti = df_train.copy()
```

```
# df_te_ti = df_test.copy()
```

In [104]:

```
df_tr_ti
```

Out[104]:

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	
0	1	CA	0.011719	1	-5.285450	26.775801	-9.527310	-12.793200	16.717800	33.737499	23.712299	-
1	1	CA	0.015625	1	-2.428420	28.430901	-9.323510	-3.757230	15.969300	30.443600	21.010300	-
2	1	CA	0.019531	1	10.671500	30.420200	15.350700	24.724001	16.143101	32.142799	25.431801	-
3	1	CA	0.023438	1	11.452500	25.609800	2.433080	12.412500	20.533300	31.494101	19.142799	-
4	1	CA	0.027344	1	7.283210	25.942600	0.113564	5.748000	19.833599	28.753599	20.572100	-
...	...	...	...	...	...	...	...	...	...	...	...	
4867416	13	SS	99.991005	1	-47.758202	-71.050400	-37.980801	-33.852901	-59.413601	-21.614300	-26.437300	-
4867417	13	SS	99.993004	0	-24.536699	79.787399	-45.435699	-6.329290	25.052500	-13.852700	23.737301	-
4867418	13	SS	99.994003	1	-42.078499	-66.937401	-27.298201	-25.931900	-47.430302	-17.464199	-20.454201	-
4867419	13	SS	99.997002	0	20.536301	10.485500	63.723400	44.422600	95.535500	76.922096	-65.109299	4
4867420	13	SS	99.998001	1	-12.230700	-39.962601	-2.675310	5.699130	-22.091000	-7.601820	6.032570	-

4867421 rows × 28 columns

In [8]:

```
df_tr_ti['experiment'] = df_tr_ti['experiment'].map({'CA': 0, 'DA': 1, 'SS': 2, 'LOFT': -1})
df_tr_ti["experiment"] = df_tr_ti["experiment"].astype('int8')
df_tr_ti
# df_te_ti['experiment'] = df_te_ti.replace({'experiment' : {'CA': 0, 'DA': 1, 'SS': 2, 'LOFT': -1
}})
```

Out[8]:

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	
0	1	0	0.011719	1	-5.285450	26.775801	-9.527310	-12.793200	16.717800	33.737499	23.712299	-
1	1	0	0.015625	1	-2.428420	28.430901	-9.323510	-3.757230	15.969300	30.443600	21.010300	-
2	1	0	0.019531	1	10.671500	30.420200	15.350700	24.724001	16.143101	32.142799	25.431801	-
3	1	0	0.023438	1	11.452500	25.609800	2.433080	12.412500	20.533300	31.494101	19.142799	-
4	1	0	0.027344	1	7.283210	25.942600	0.113564	5.748000	19.833599	28.753599	20.572100	-
...	...	...	...	...	...	...	...	...	...	...	...	
4867416	13	2	99.991005	1	-47.758202	-71.050400	-37.980801	-33.852901	-59.413601	-21.614300	-26.437300	-
4867417	13	2	99.993004	0	-24.536699	79.787399	-45.435699	-6.329290	25.052500	-13.852700	23.737301	-
4867418	13	2	99.994003	1	-42.078499	-66.937401	-27.298201	-25.931900	-47.430302	-17.464199	-20.454201	-
4867419	13	2	99.997002	0	20.536301	10.485500	63.723400	44.422600	95.535500	76.922096	-65.109299	4

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	
4867420	13	2	99.998001	1	12.230700	39.962601	-2.675310	5.699130	22.091000	-7.601820	6.032570	-

4867421 rows × 28 columns

In [9]:

```
df_tr_ti['event'] = df_tr_ti['event'].map({'A': -1, 'B': 2, 'C': 0, 'D': 1})
df_tr_ti["event"] = df_tr_ti["event"].astype('int8')
df_tr_ti
# df_tr_ti['event'] = df_te_ti.replace({'event' : {'A': -1, 'B': 2, 'C': 0, 'D': 1}})
```

Out[9]:

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	
0	1	0	0.011719	1	-5.285450	26.775801	-9.527310	-12.793200	16.717800	33.737499	23.712299	-
1	1	0	0.015625	1	-2.428420	28.430901	-9.323510	-3.757230	15.969300	30.443600	21.010300	-
2	1	0	0.019531	1	10.671500	30.420200	15.350700	24.724001	16.143101	32.142799	25.431801	-
3	1	0	0.023438	1	11.452500	25.609800	2.433080	12.412500	20.533300	31.494101	19.142799	-
4	1	0	0.027344	1	7.283210	25.942600	0.113564	5.748000	19.833599	28.753599	20.572100	-
...	...	...	...	...	...	...	...	...	...	...	...	.
4867416	13	2	99.991005	1	-47.758202	-71.050400	-37.980801	-33.852901	-59.413601	-21.614300	-26.437300	-
4867417	13	2	99.993004	0	-24.536699	79.787399	-45.435699	-6.329290	25.052500	-13.852700	23.737301	-
4867418	13	2	99.994003	1	-42.078499	-66.937401	-27.298201	-25.931900	-47.430302	-17.464199	-20.454201	-
4867419	13	2	99.997002	0	20.536301	10.485500	63.723400	44.422600	95.535500	76.922096	-65.109299	4
4867420	13	2	99.998001	1	-12.230700	-39.962601	-2.675310	5.699130	-22.091000	-7.601820	6.032570	-

4867421 rows × 28 columns

Now we will structure the data such as we can study the features for each pilot with time.

- Since we now in data we have multiple crews and each crew has a pilot on seat. So we will break the data such that we can access the data for each pilot or seat for each crew and do the analysis of this data with each event over time.

In [10]:

```
df_tr_ti = df_tr_ti.set_index(['crew', 'seat', 'experiment'])
# df_te_ti = df_te_ti.set_index(['crew', 'seat', 'experiment'])
```

In [11]:

```
#sorting values by increasing time
df_tr_ti = df_tr_ti.sort_values(by='time').sort_index()
df_tr_ti
```

Out[11]:

			time	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2
crew	**seat**	**experiment**									
1	**0**	**0**	0.183594	1.993100	8.099320	8.89996	-11.851900	0.637441	-5.892530	11.845500	12.453800

			time	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2
crew	seat	experiment									
		0	0.187500	1.442160	7.208550	6.153520	-18.372601	7.036370	-2.301950	17.986099	10.608600
		0	0.191406	1.881800	1.906450	-7.29714	0.054378	9.325300	-10.387200	5.171800	-2.439820
		0	0.195312	3.100600	0.000000	8.11878	10.091100	4.783260	-14.553400	2.925110	2.500380
		0	0.199219	0.000000	-0.260347	5.00783	10.015900	-1.802200	-7.049790	8.763830	7.066740
...	...	...	...	...	...	...	...	...	...	...	...
13	1	2	359.796021	71.033798	72.598602	-4.68886	-3.673780	33.645302	23.406900	42.619400	42.489899
		2	359.801025	74.047302	75.331100	-2.15364	5.039420	39.655499	14.007300	28.258699	44.120701
		2	359.806030	67.345299	68.202003	-5.17588	-0.318111	30.432100	7.881610	18.732300	37.080502
		2	359.811005	73.157303	72.831802	1.49136	-2.395910	33.541698	29.011000	38.554501	45.870399
		2	359.816010	81.068604	80.584099	11.51280	9.592110	44.053299	44.891998	51.497002	52.224899

4867421 rows × 25 columns

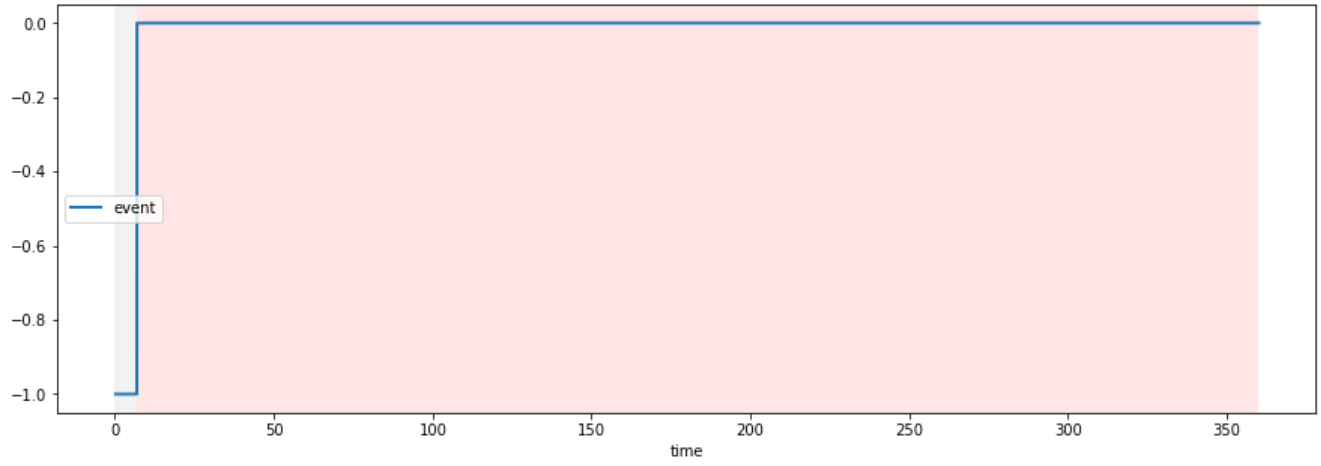
In [26]:

```
#{'CA': 0, 'DA': 1, 'SS': 2, 'LOFT': -1}
#{'A': -1, 'B': 2, 'C': 0, 'D': 1}
# 0->CA,1->DA ,2->SS,-1->LOFT
colors = {0: 'red', 1: 'green', 2: 'blue', -1: 'gray'}
def get_plot(data,feature, experiment):

  #getting data for the experiment for single pilot in a single crew
  temp = data.loc[experiment]
  ax = temp.plot(kind='line',x='time',y=feature,figsize=(15,5),linewidth=2)

  #setting initial value of time=0 and last value as the maximum value in our data rest between is
from change
  time_range = [0] + list(temp.time.values) + [temp.time.max()]

  #Setting the initial event as per data and rest is from our change
  events = [temp.event.iloc[0]] + list(temp.event.values)

  for i in range(len(time_range)-1):
    #plotting different events with different colors

    ax.axvspan(time_range[i], time_range[i+1], facecolor=colors[events[i]], alpha=0.1)

  plt.show()
```

Studying data for a single pilot(crew 1 and seat 0) in from a single crew over time of experiments for each event

In [ ]:

```
get_plot(df_tr_ti.loc[1,0],'event',0)
```

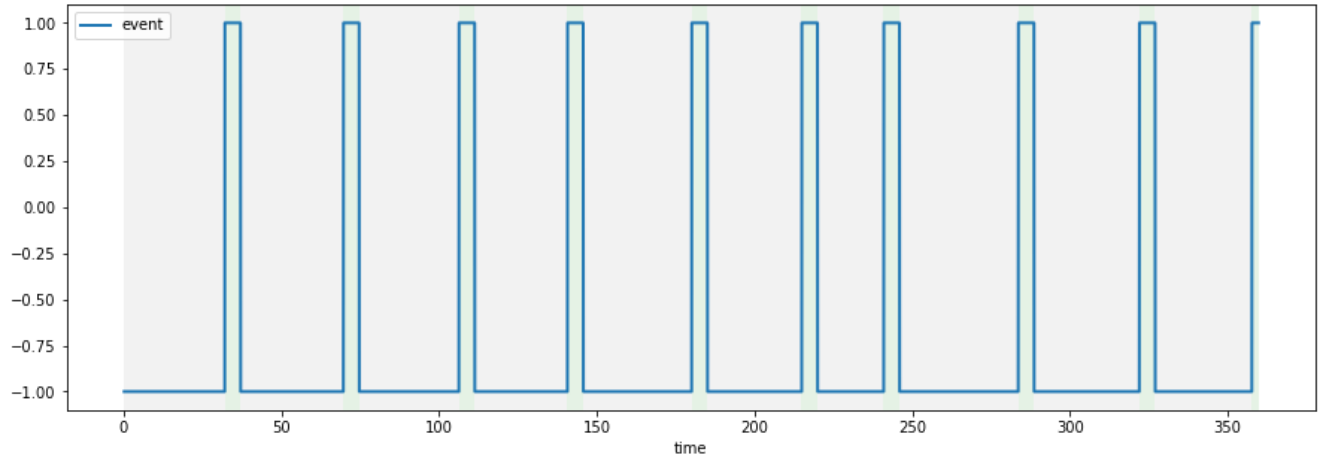


Observations:

- From above graph we can see event 0 i.e CA is happening for all timeline.

In [ ]:

```
get_plot(df_tr_ti.loc[1,0],'event',1)
```

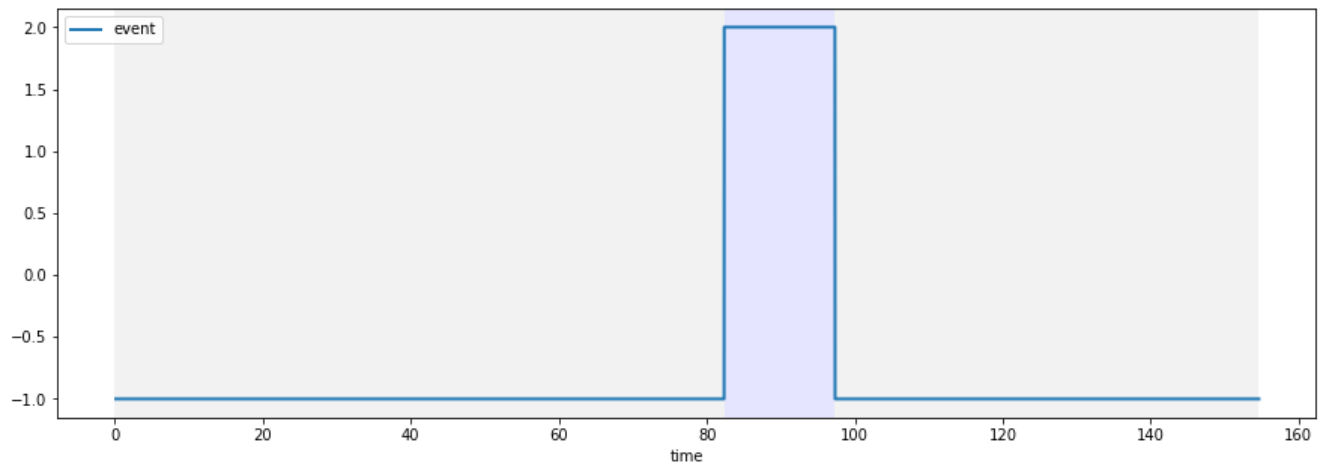


Observations:

- Here we can see event DA is happening in alternate regular intervals over the period of time.

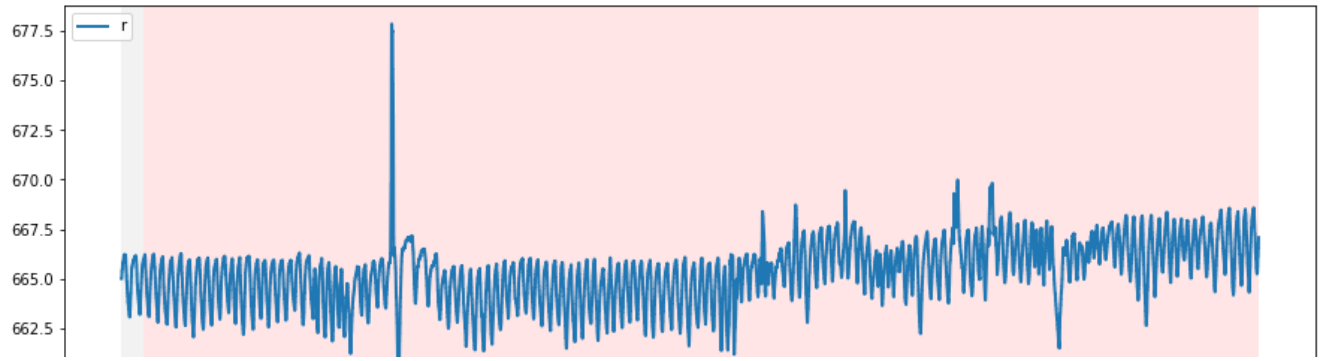
In [ ]:

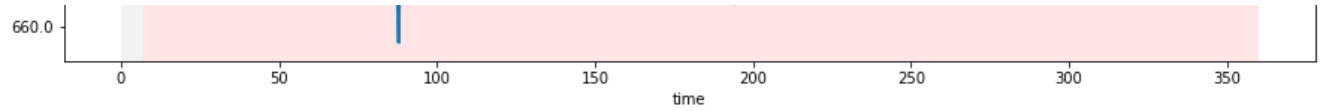
```
get_plot(df_tr_ti.loc[1,0],'event',2)
```



In [ ]:

```
get_plot(df_tr_ti.loc[1,0],'r',0)
```



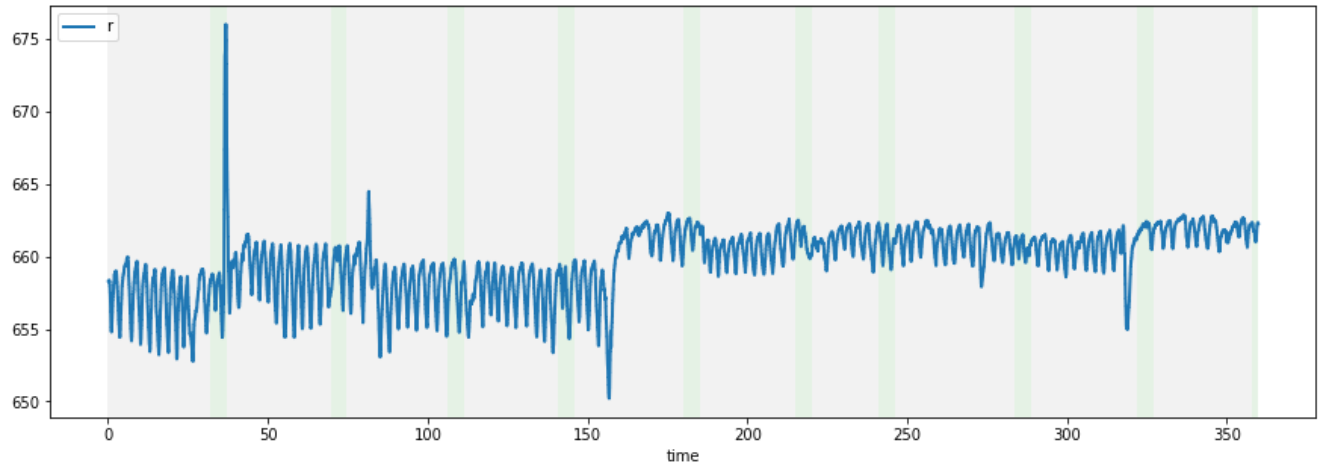


Observations:

- The respiration is almost same except at range around 80 where we can see a sharp rise in range.

In [ ]:

```
get_plot(df_tr_ti.loc[1,0],'r',1)
```

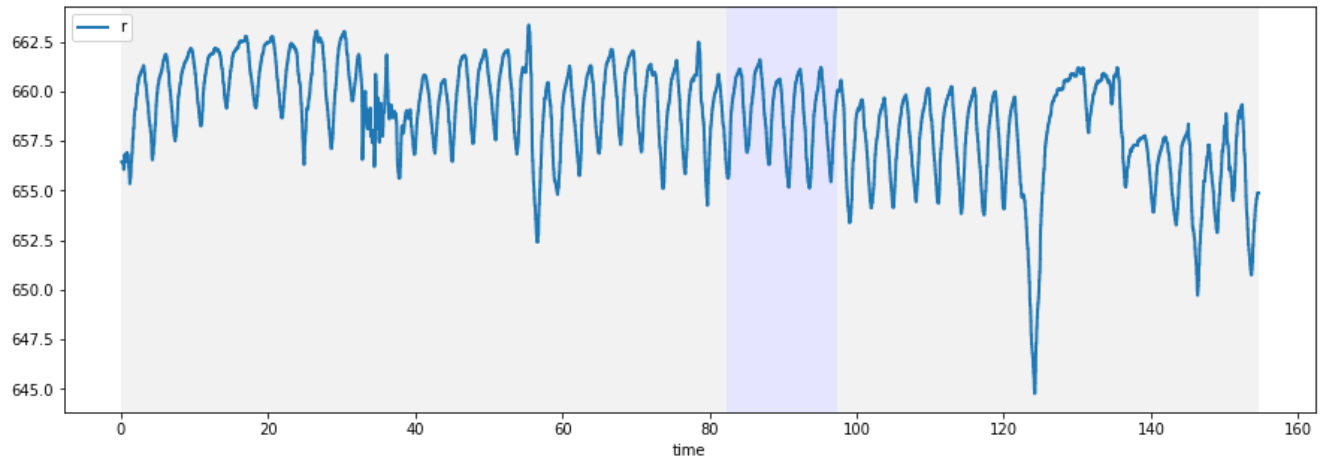


Observations:

- Apart from first occurance of event DA rest of the plot is almost constant except a little change in range of values.

In [ ]:

```
get_plot(df_tr_ti.loc[1,0],'r',2)
```

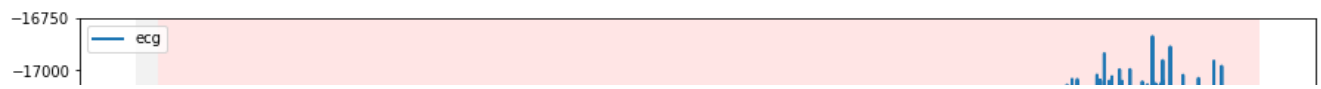


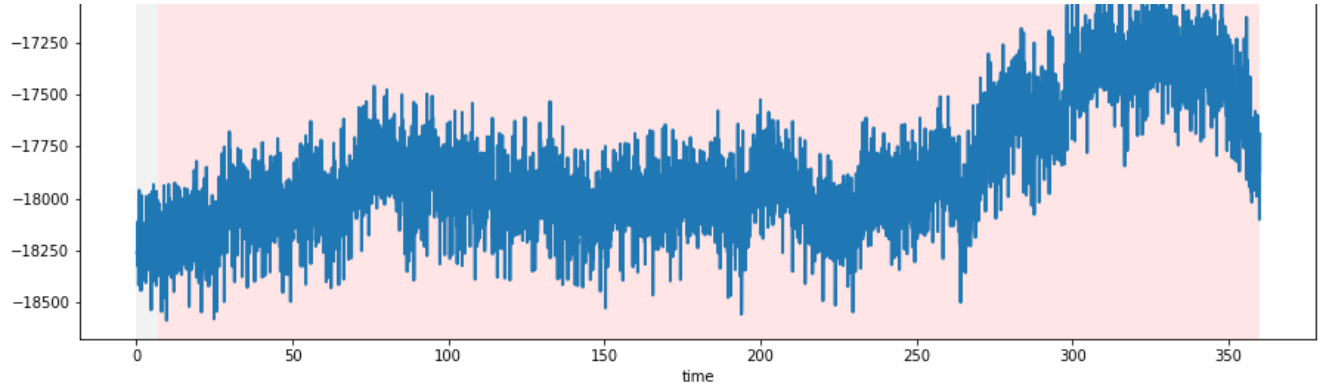
Observations:

- A sharp dip can be observed near the end of the timeline in respiration range although it remained constant for most of the part

In [ ]:

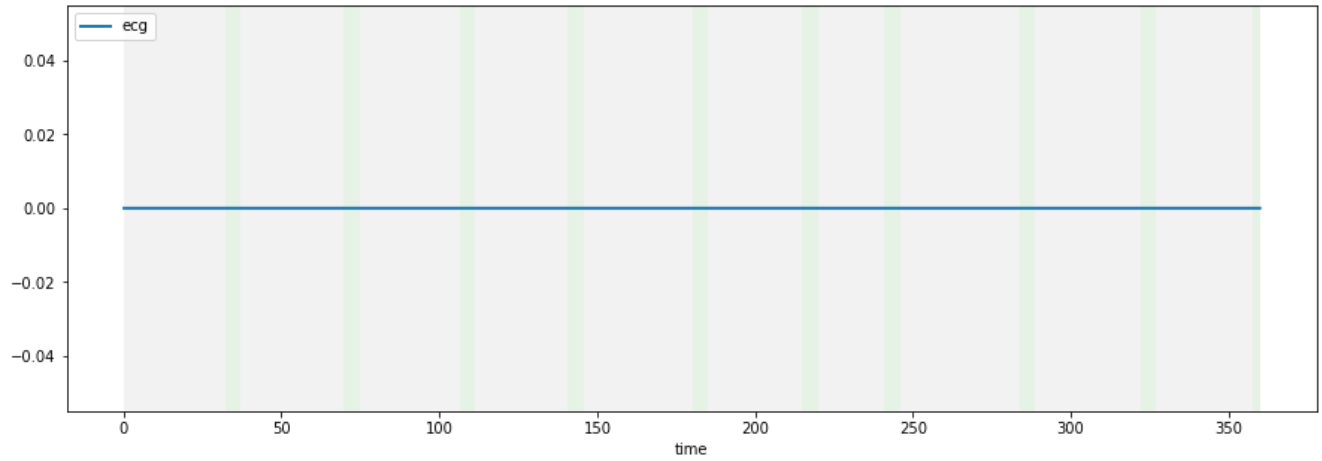
```
get_plot(df_tr_ti.loc[1,0],'ecg',0)
```





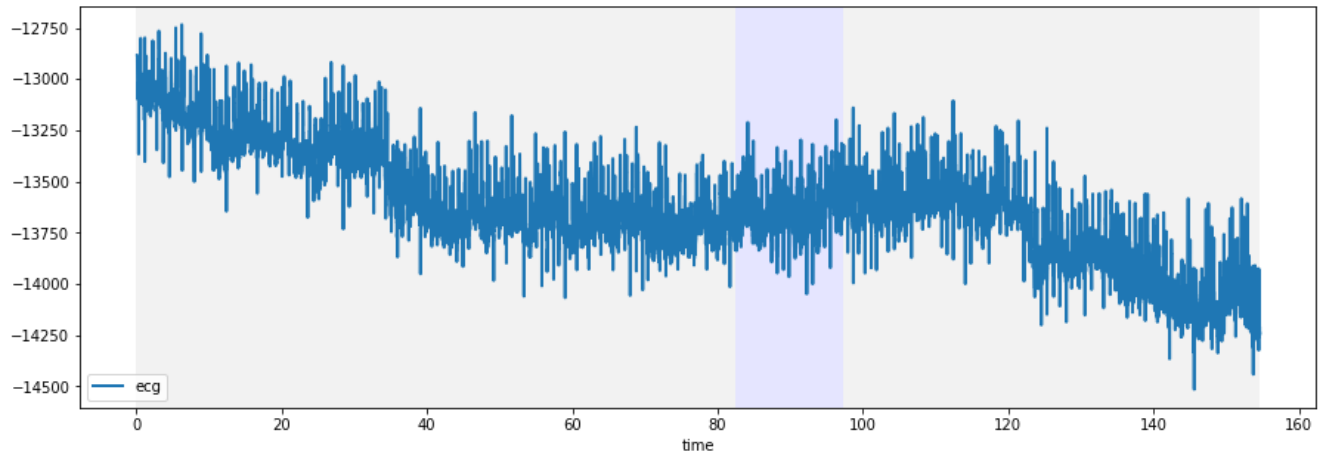
In [ ]:

```
get_plot(df_tr_ti.loc[1,0],'ecg',1)
```



In [ ]:

```
get_plot(df_tr_ti.loc[1,0],'ecg',2)
```

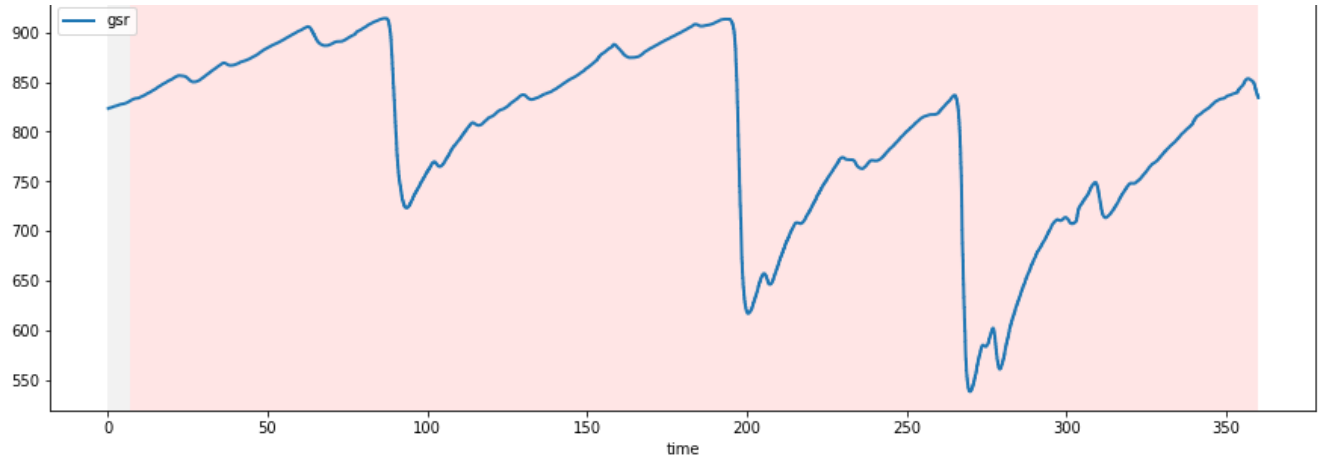


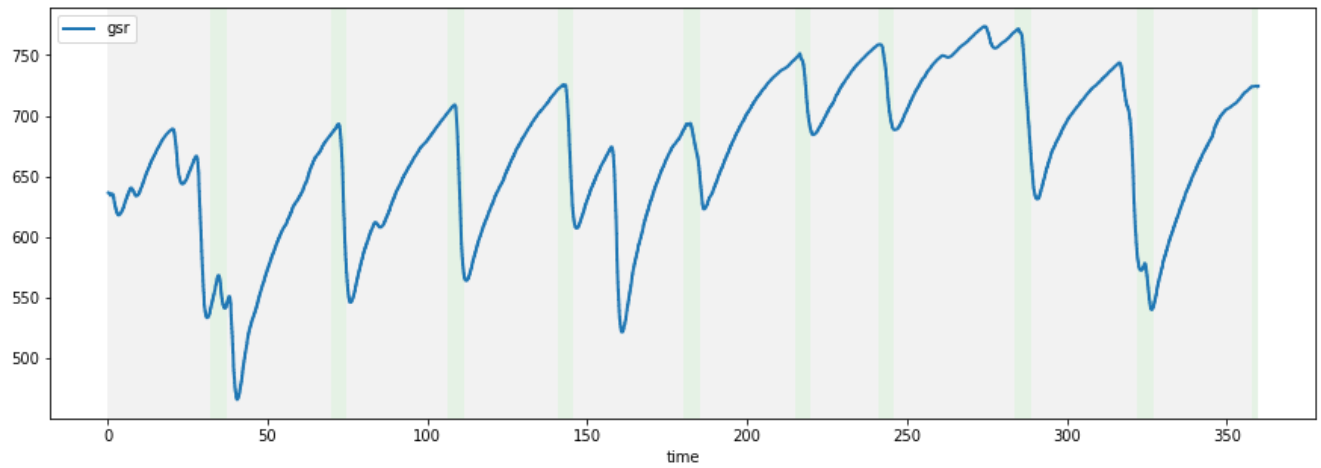
Observations:

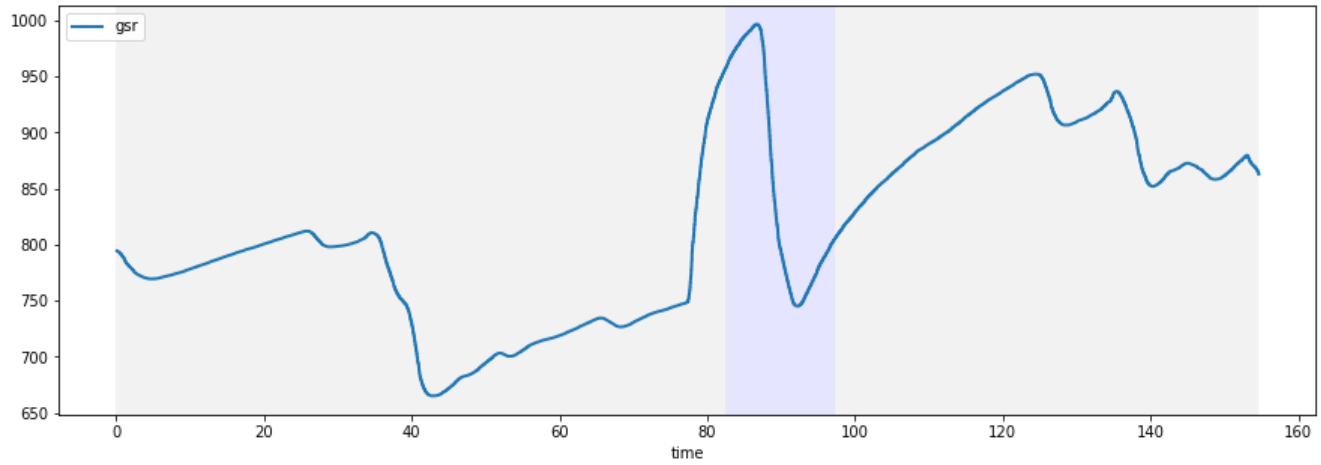
- Except for the event DA the ecg for CA event has a incline and for the SS event it has decline.
- It is interesting to observe that ecg has not given any reading for event DA. We will verify this part in another pilot's data.

In [ ]:

```
get_plot(df_tr_ti.loc[1,0],'gsr',0)
get_plot(df_tr_ti.loc[1,0],'gsr',1)
get_plot(df_tr_ti.loc[1,0],'gsr',2)
```





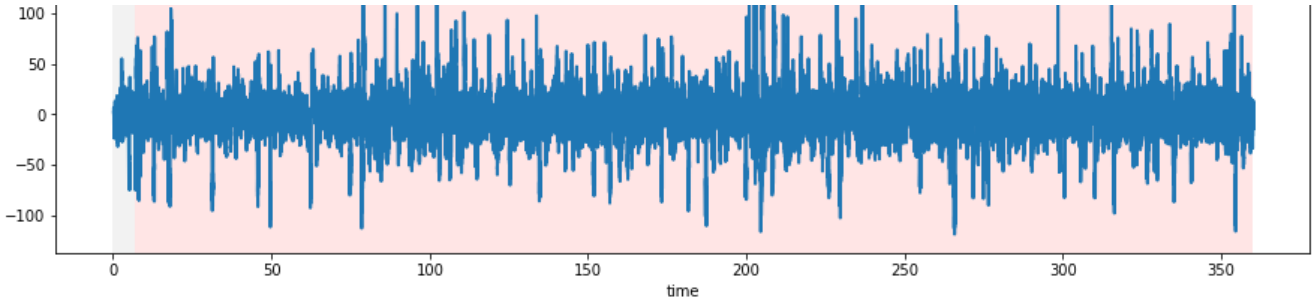


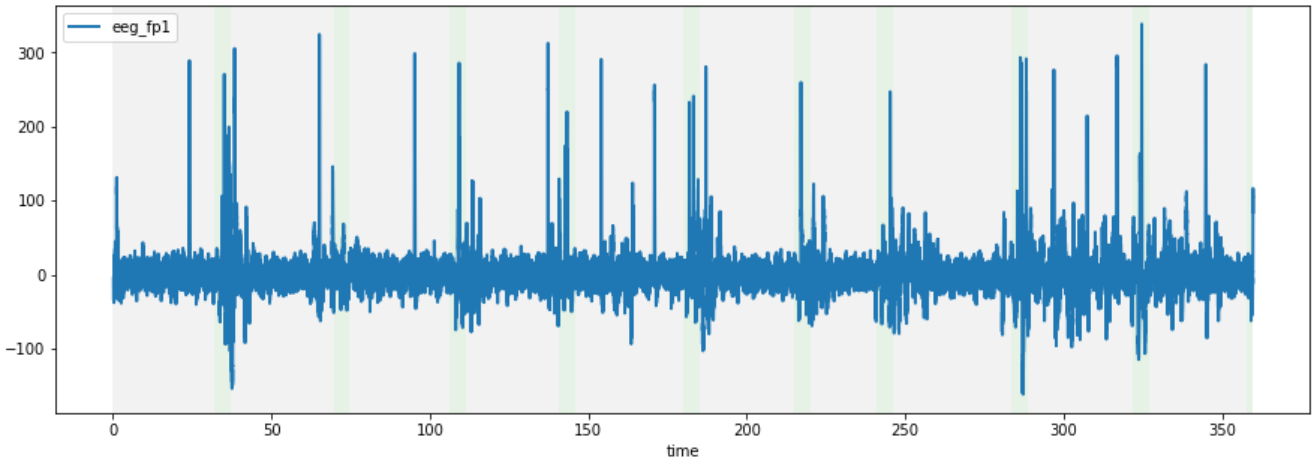
Observations:

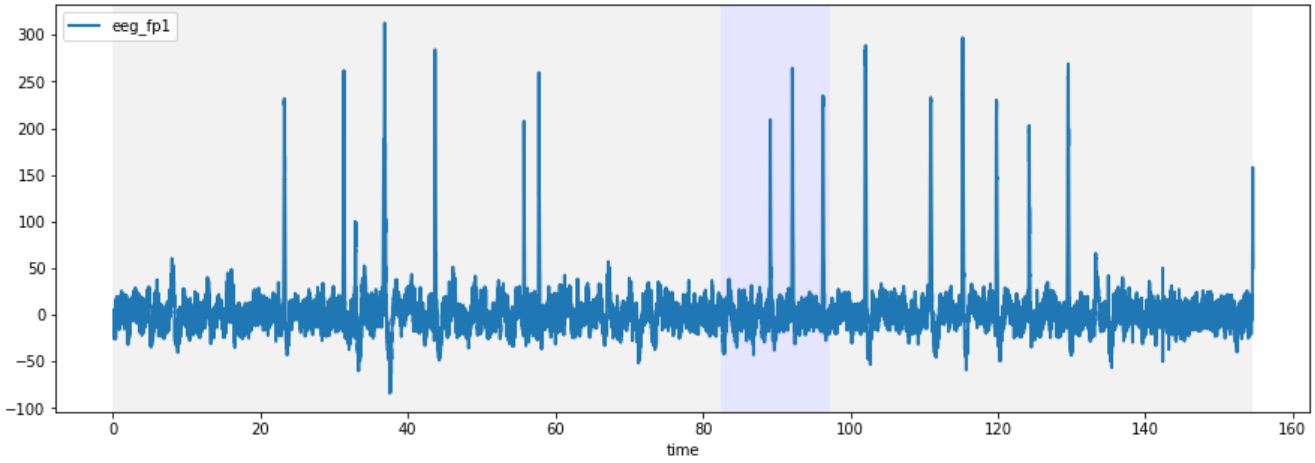
- For event CA we can see a periodicity of peaks throughout.
- For even DA we can see a peak almost every time the event is happening.
- For event SS it is easily visible a sharp spike.

In [ ]:

```
get_plot(df_tr_ti.loc[1,0],'eeg_fp1',0)
get_plot(df_tr_ti.loc[1,0],'eeg_fp1',1)
get_plot(df_tr_ti.loc[1,0],'eeg_fp1',2)
```

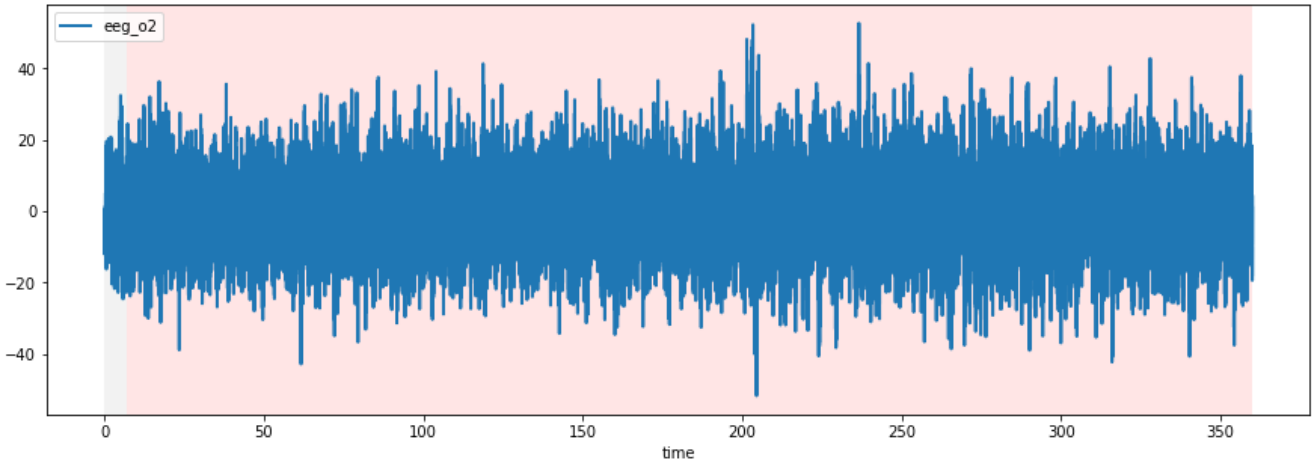


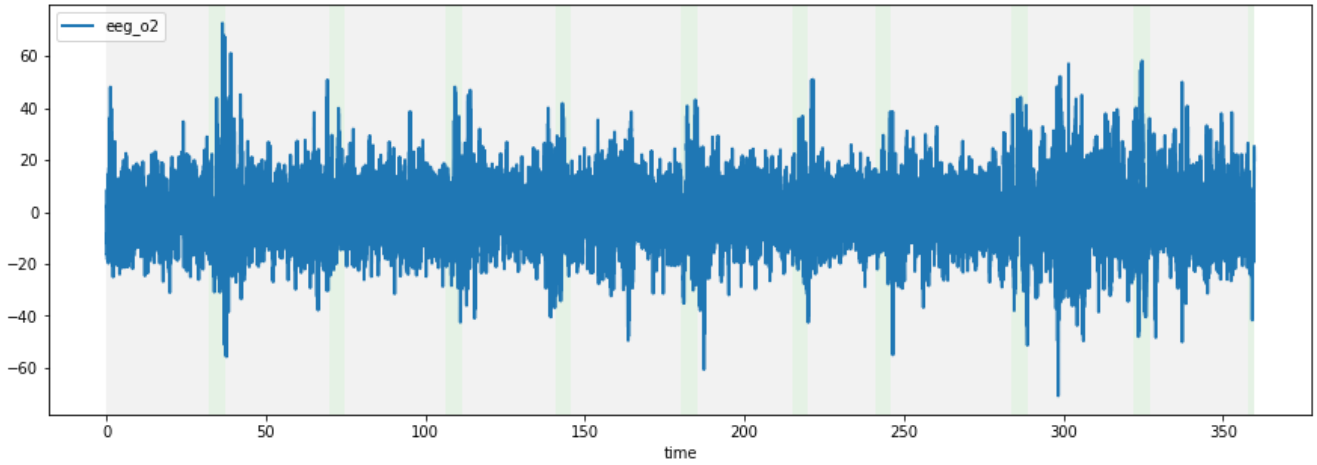


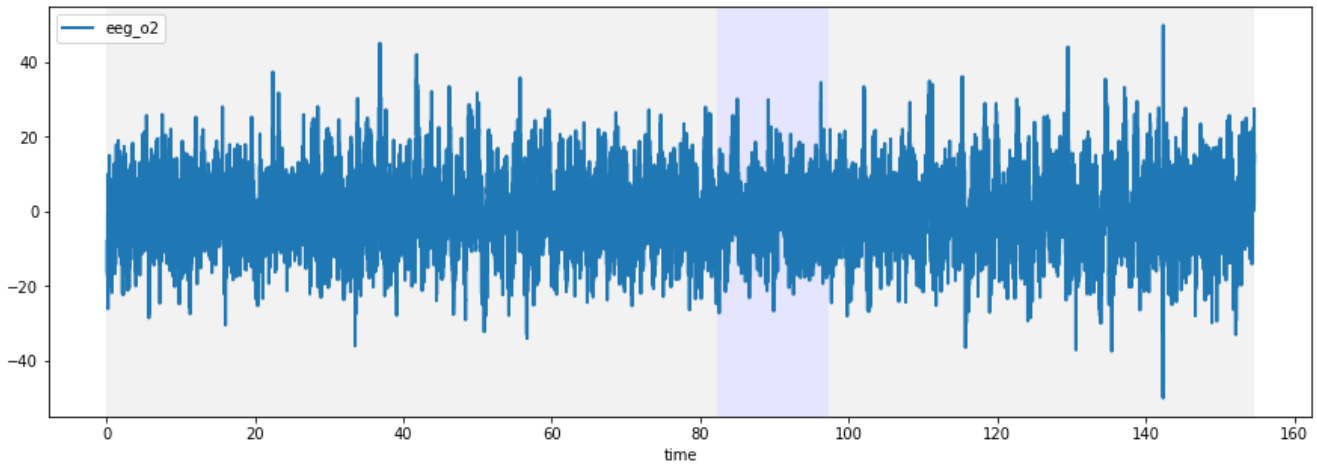


In [ ]:

```
get_plot(df_tr_ti.loc[1,0],'eeg_o2',0)
get_plot(df_tr_ti.loc[1,0],'eeg_o2',1)
get_plot(df_tr_ti.loc[1,0],'eeg_o2',2)
```







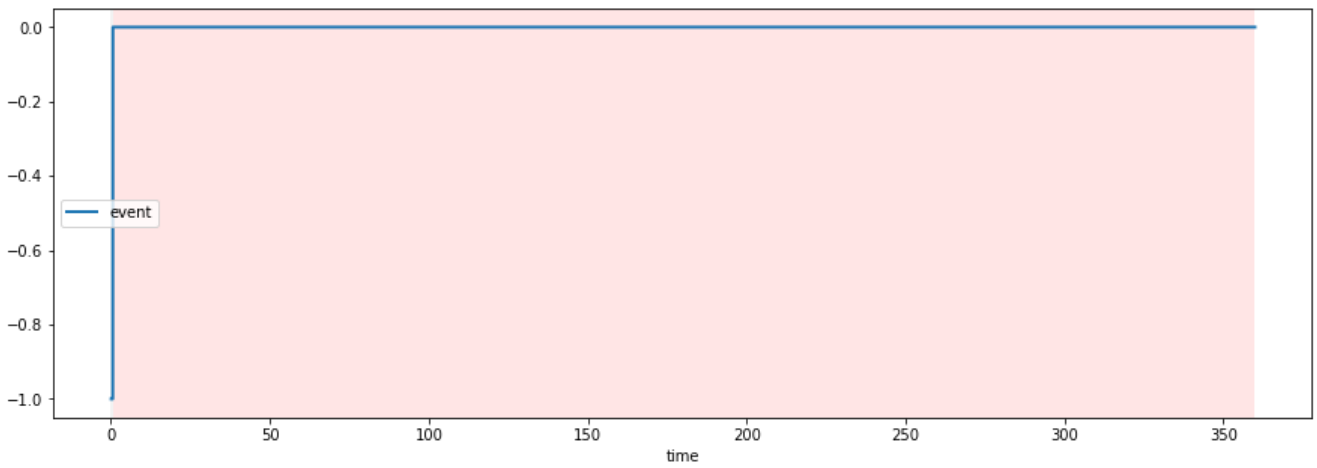
Observations:

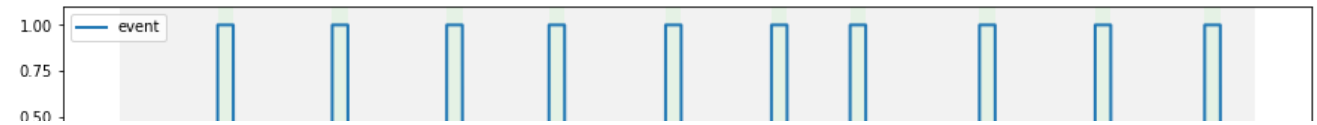
- From eeg we have only plotted 2 features one thing we can notice is the range of each eeg feature differes for each experiment.

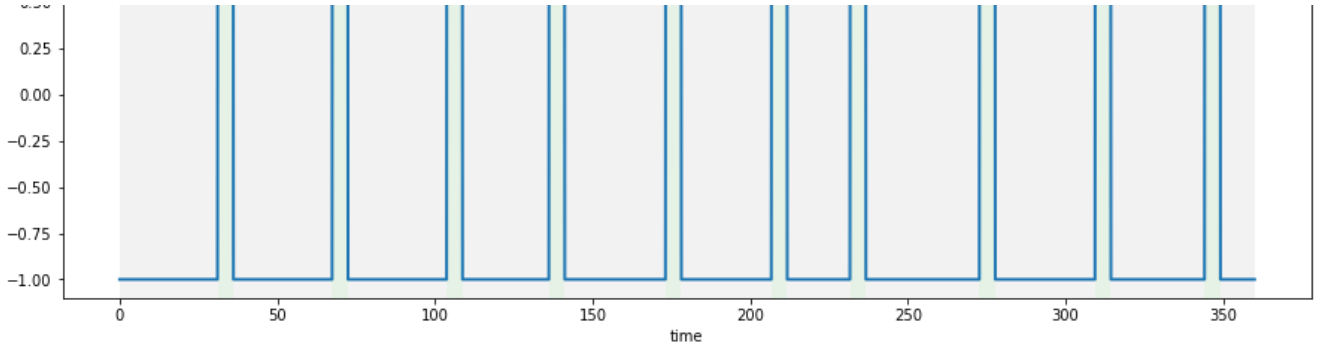
Lets take a look at data of one more pilot from crew 4 and seat 1

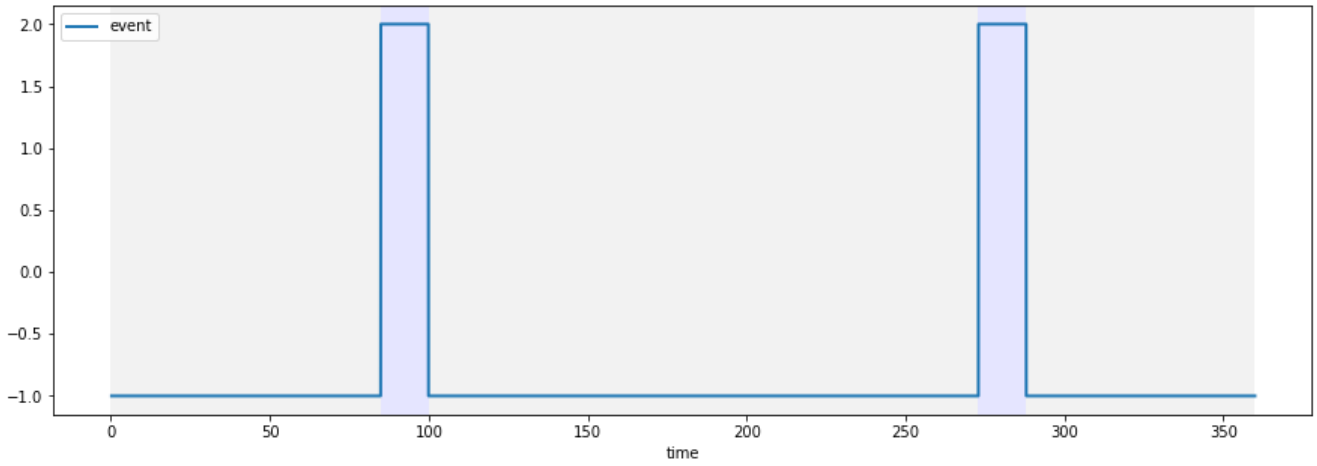
In [ ]:

```
get_plot(df_tr_ti.loc[4,1],'event',0)
get_plot(df_tr_ti.loc[4,1],'event',1)
get_plot(df_tr_ti.loc[4,1],'event',2)
```



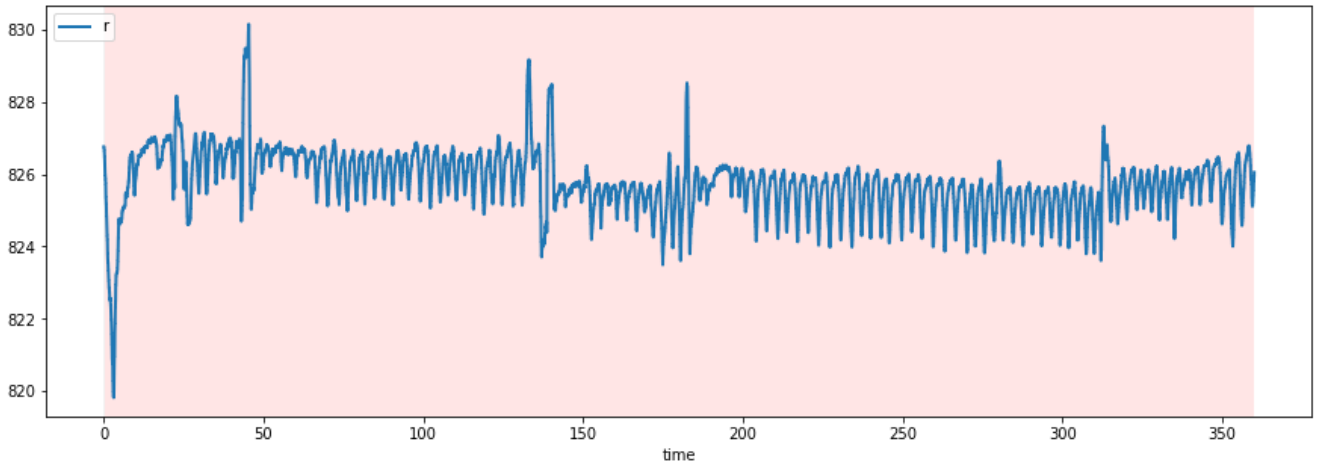


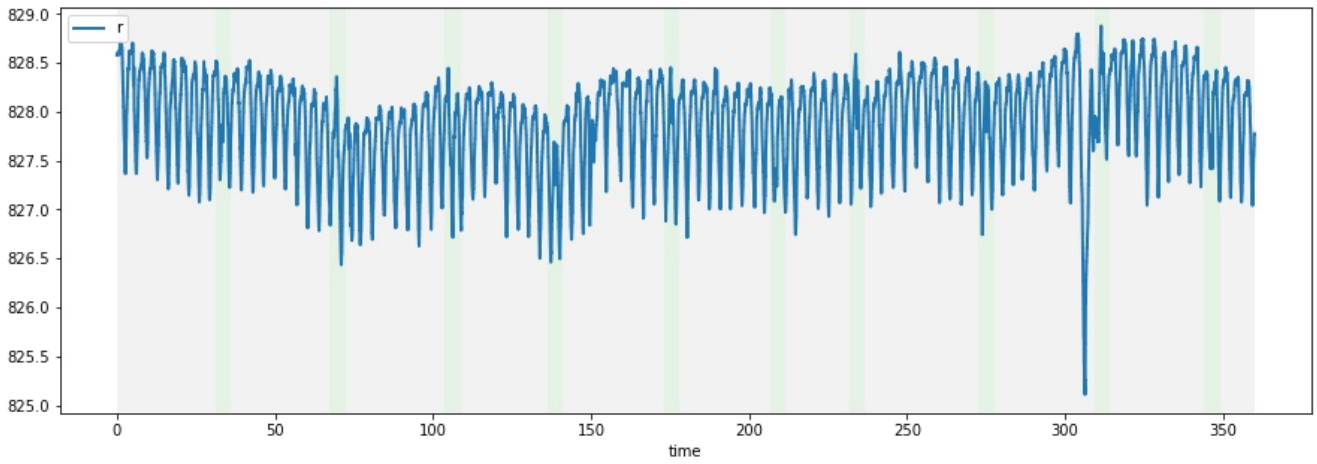


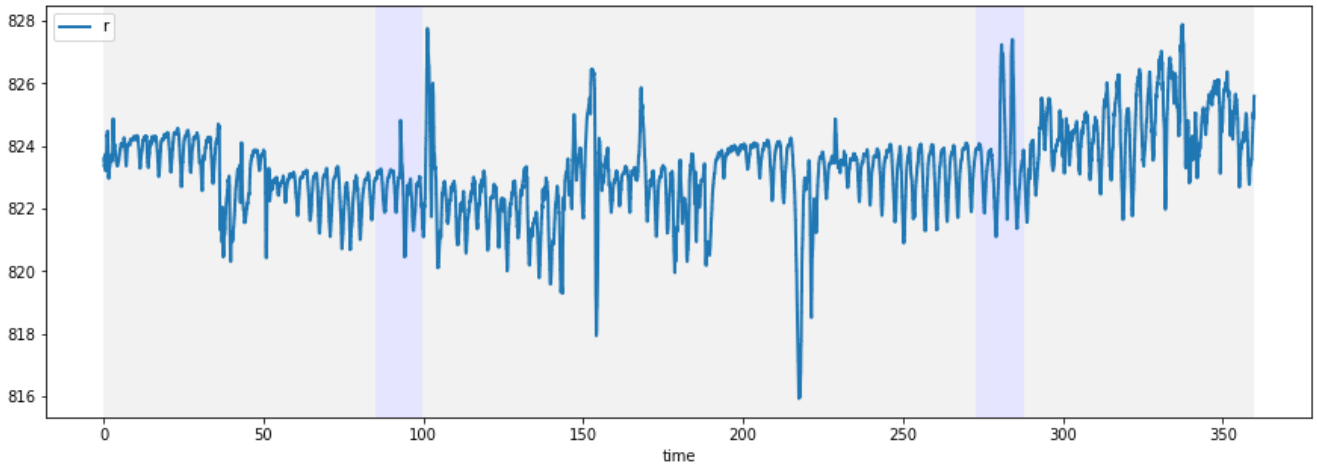


In [ ]:

```
get_plot(df_tr_ti.loc[4,1],'r',0)
get_plot(df_tr_ti.loc[4,1],'r',1)
get_plot(df_tr_ti.loc[4,1],'r',2)
```

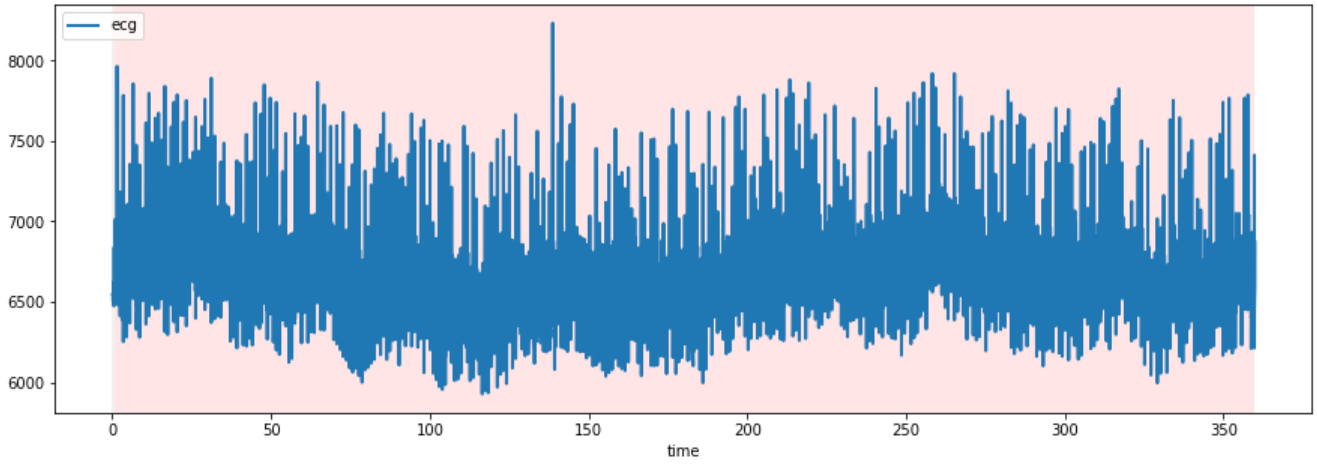


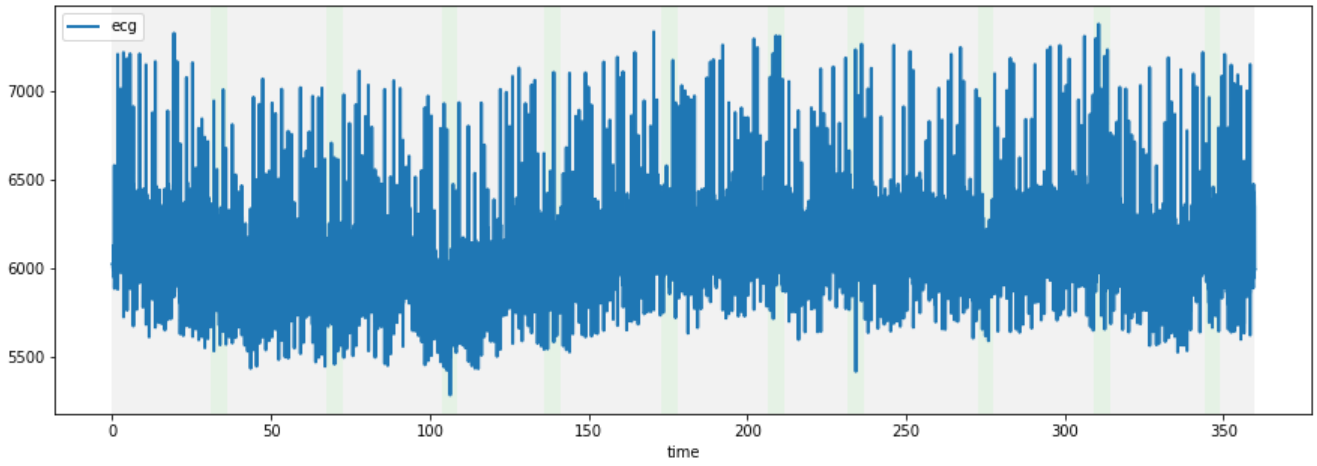


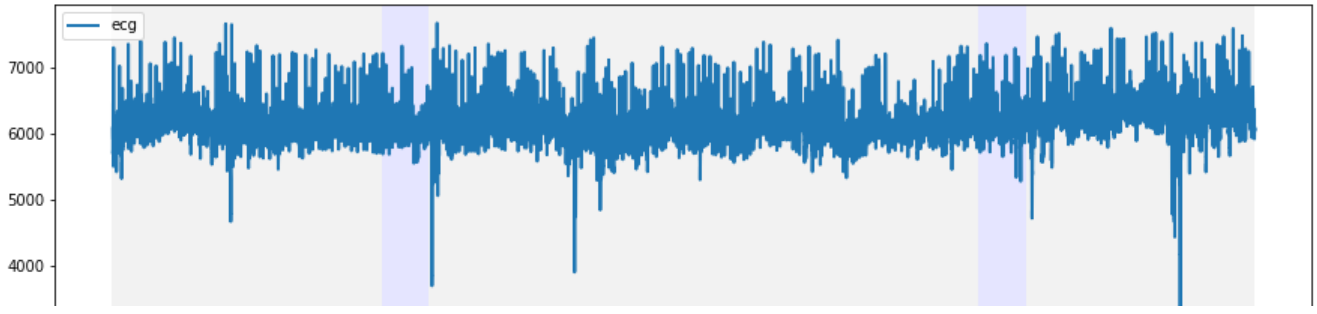


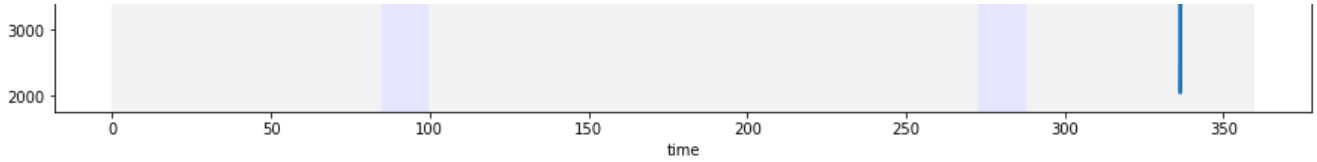
In [ ]:

```
get_plot(df_tr_ti.loc[4,1],'ecg',0)
get_plot(df_tr_ti.loc[4,1],'ecg',1)
get_plot(df_tr_ti.loc[4,1],'ecg',2)
```



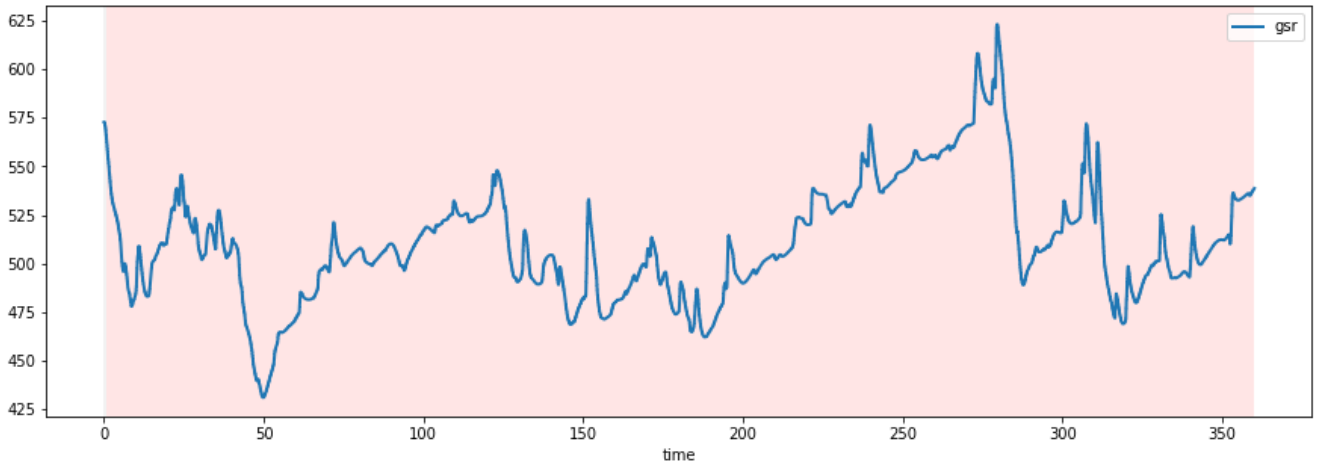


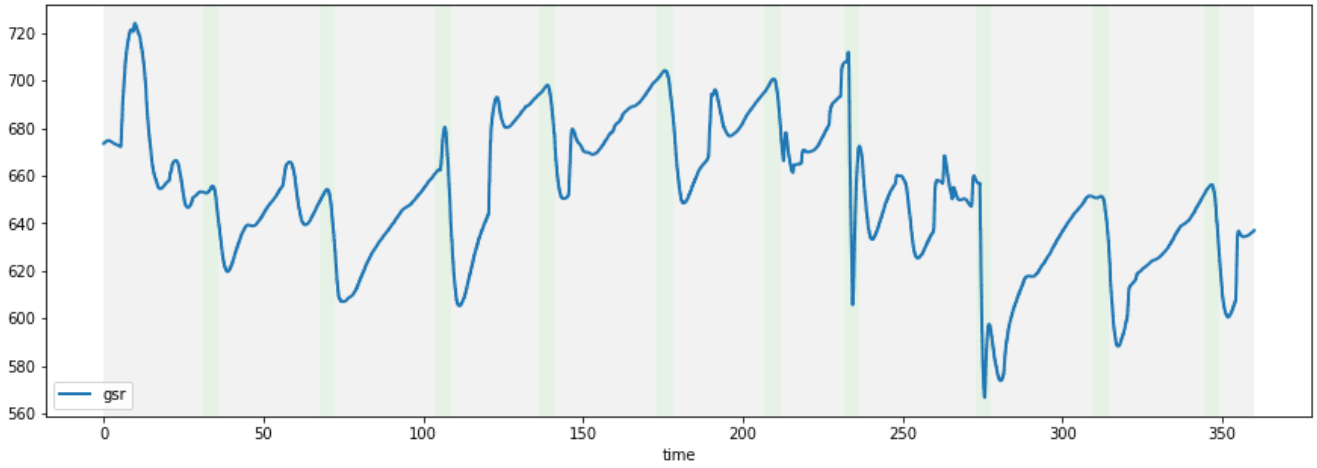


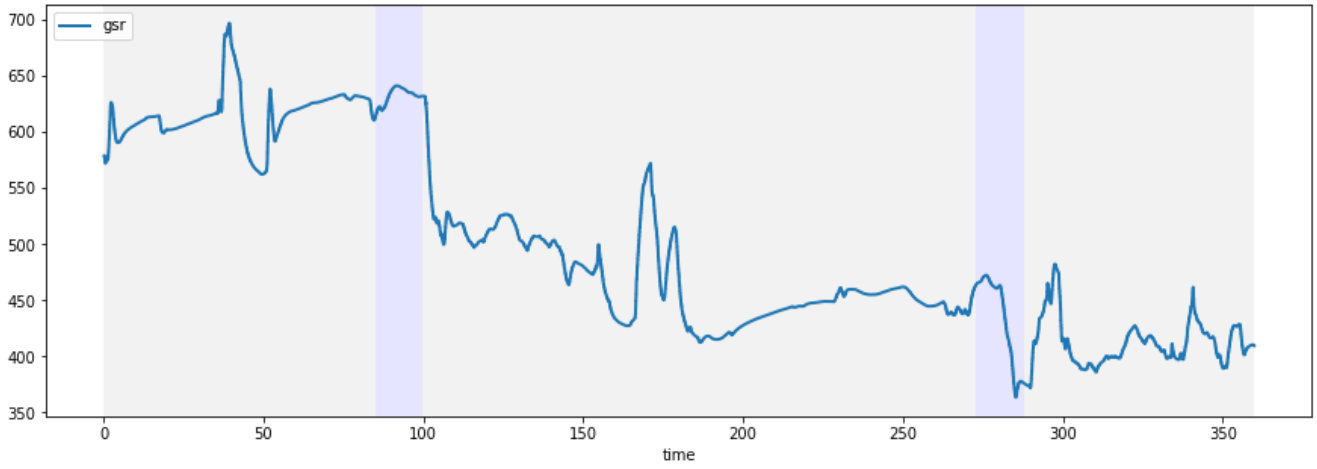


In [ ]:

```
get_plot(df_tr_ti.loc[4,1],'gsr',0)
get_plot(df_tr_ti.loc[4,1],'gsr',1)
get_plot(df_tr_ti.loc[4,1],'gsr',2)
```

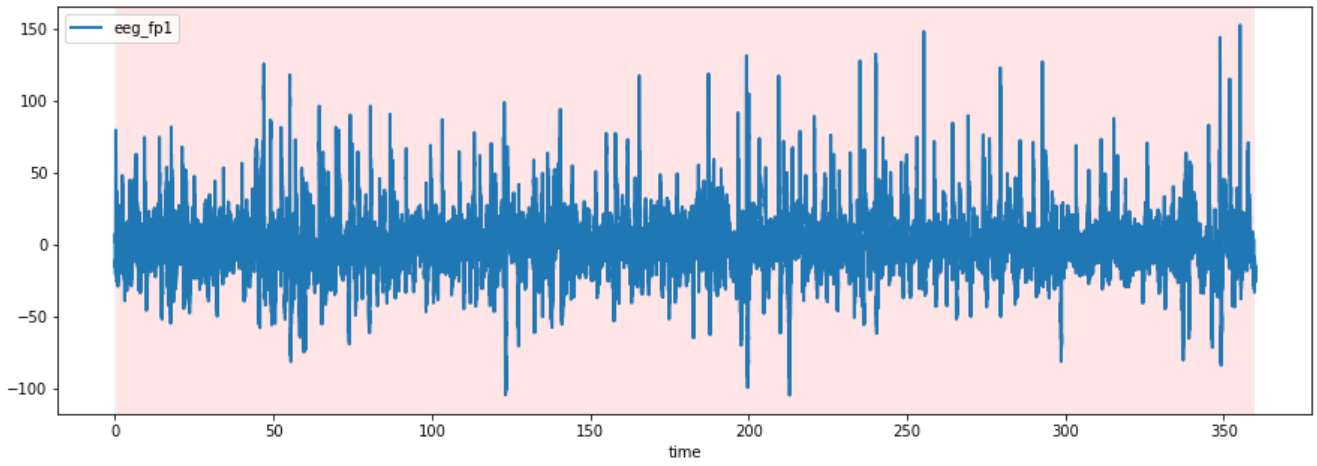


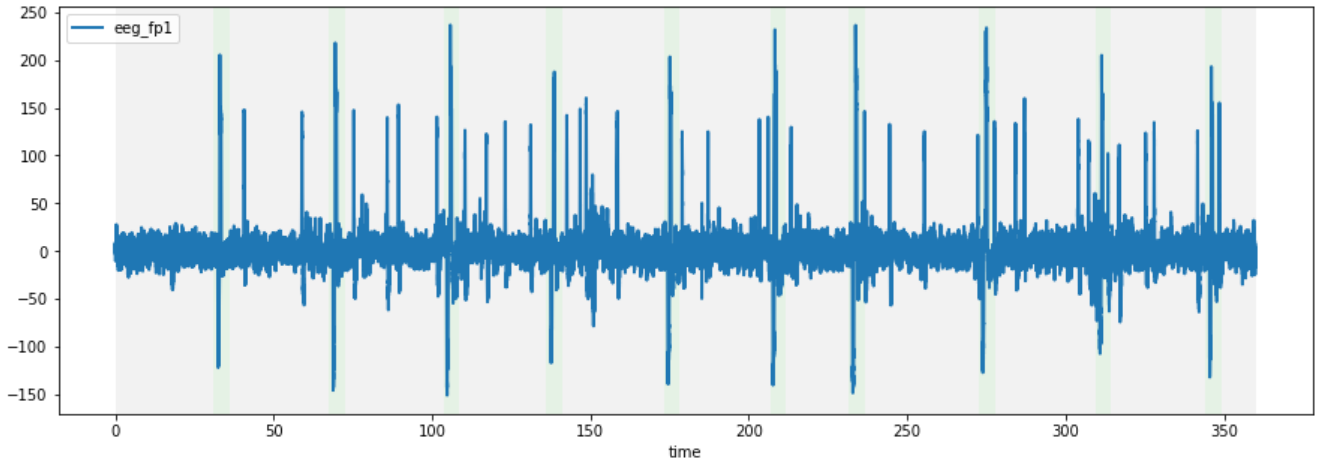


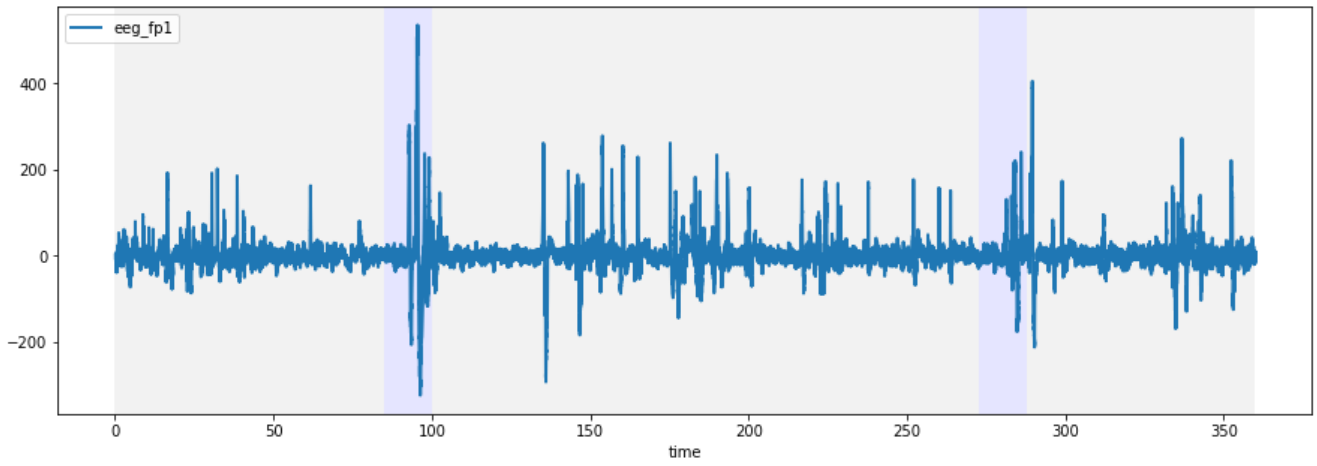


In [ ]:

```
get_plot(df_tr_ti.loc[4,1],'eeg_fp1',0)
get_plot(df_tr_ti.loc[4,1],'eeg_fp1',1)
get_plot(df_tr_ti.loc[4,1],'eeg_fp1',2)
```

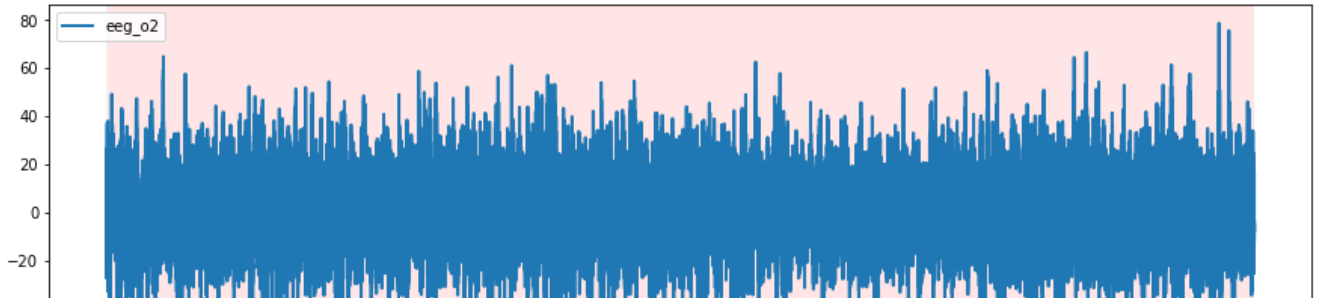


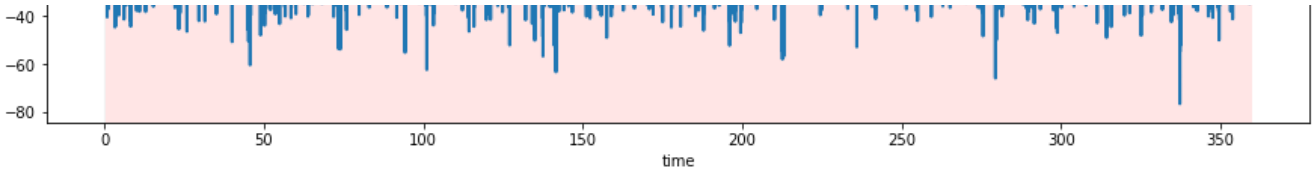


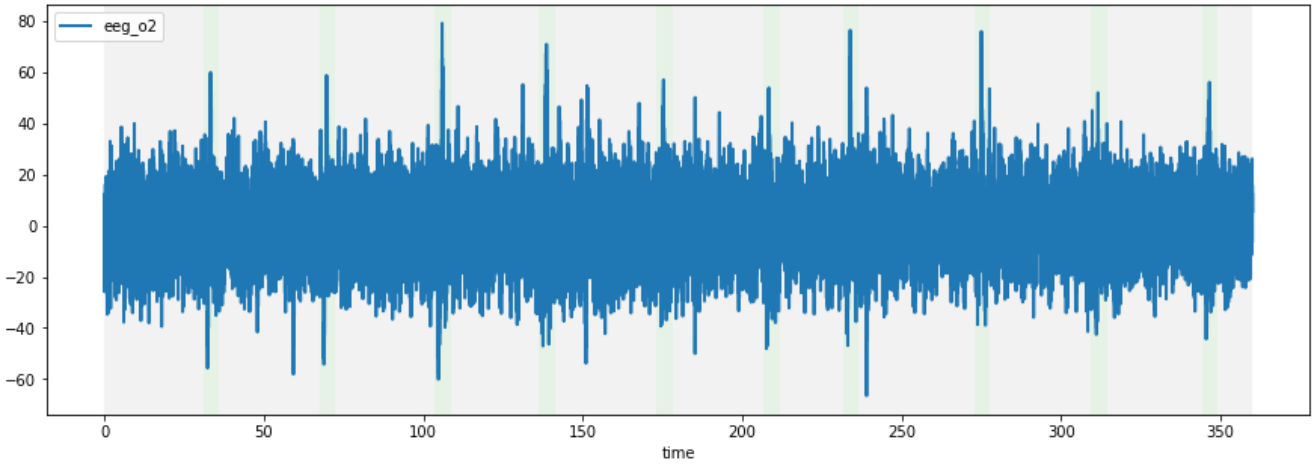


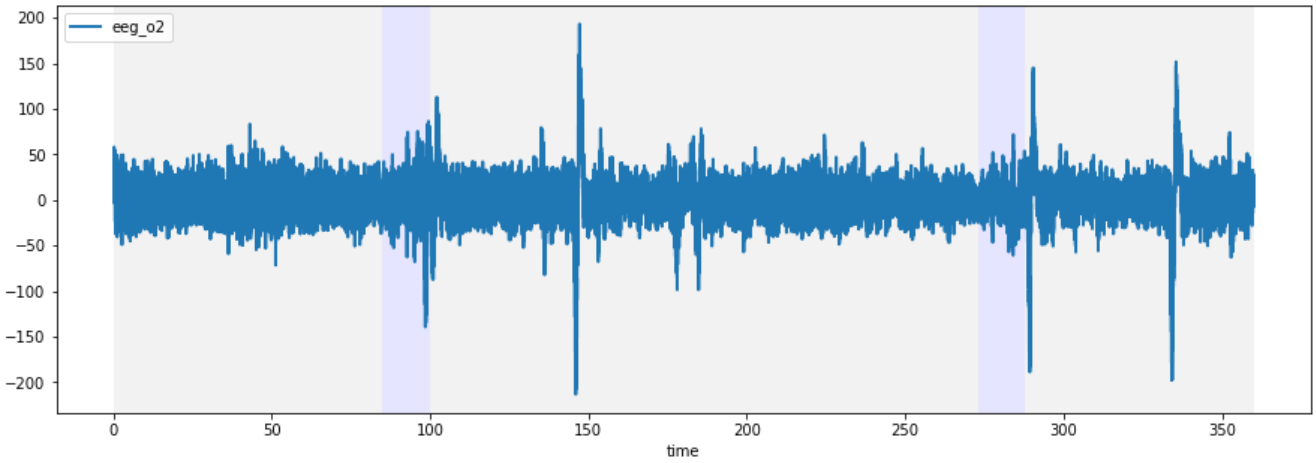
In [ ]:

```
get_plot(df_tr_ti.loc[4,1],'eeg_o2',0)
get_plot(df_tr_ti.loc[4,1],'eeg_o2',1)
get_plot(df_tr_ti.loc[4,1],'eeg_o2',2)
```









Observations:

- So all in all we can say that all the eeg features combined will develop some patterns to detect event.
- The patterns of this pilot in most cases is quite similar to the first one

Calculating rolling values

In [39]:

```
def calculate_values(data1,data2):
  feat = ['eeg_fp1', 'eeg_f7', 'eeg_f8', 'eeg_t4', 'eeg_t6', 'eeg_t5','eeg_t3', 'eeg_fp2', 'eeg_o1'
, 'eeg_p3', 'eeg_pz', 'eeg_f3', 'eeg_fz','eeg_f4', 'eeg_c4', 'eeg_p4', 'eeg_poz', 'eeg_c3', 'eeg_cz
', 'eeg_o2','ecg', 'r', 'gsr']
  feat2 = ['event']

  x = PrettyTable()
  x.field_names = ['Value'] + feat
  mn = ['Mean']
  med = ['Median']
  vr = ['Variance']
  mx = ['Max']
  mini = ['Min']
  tfq = ['25th percentile']
  sfq = ['75th percentile']
  iqr = ['IQR']
  md = ['Mode']


  y = PrettyTable()
```

```
  y.field_names = ['Value'] + feat
  mn2 = ['Mean']
  med2 = ['Median']
  vr2 = ['Variance']
  mx2 = ['Max']
  mini2 = ['Min']
  tfq2 = ['25th percentile']
  sfq2 = ['75th percentile']
  iqr2 = ['IQR']
  md2 = ['Mode']

  w = PrettyTable()
  w.field_names = ['Value'] + feat2

  z = PrettyTable()
  z.field_names = ['Value'] + feat2

  for f in feat:
    mn.append(data1[f].mean())
    med.append(data1[f].median())
    vr.append(data1[f].var())
    mx.append(data1[f].max())
    mini.append(data1[f].min())
    tfq.append(data1[f].quantile(0.25))
    sfq.append(data1[f].quantile(0.75))
    iqr.append(data1[f].quantile(0.75) - data1[f].quantile(0.25))

    mn2.append(data2[f + '_roll_mean'].mean())
    med2.append(data2[f + '_roll_mean'].median())
    vr2.append(data2[f + '_roll_mean'].var())
    mx2.append(data2[f + '_roll_mean'].max())
    mini2.append(data2[f + '_roll_mean'].min())
    tfq2.append(data2[f + '_roll_mean'].quantile(0.25))
    sfq2.append(data2[f + '_roll_mean'].quantile(0.75))
    iqr2.append(data2[f + '_roll_mean'].quantile(0.75) - data2[f +'_roll_mean'].quantile(0.25))

  for f in feat2:
    md.append(data1[f].mode())
    md2.append(data2[f].mode())


  x.add_row(mn)
  x.add_row(med)
  x.add_row(vr)
  x.add_row(mx)
  x.add_row(mini)
  x.add_row(tfq)
  x.add_row(sfq)
  x.add_row(iqr)
  w.add_row(md)

  y.add_row(mn2)
  y.add_row(med2)
  y.add_row(vr2)
  y.add_row(mx2)
  y.add_row(mini2)
  y.add_row(tfq2)
  y.add_row(sfq2)
  y.add_row(iqr2)
  z.add_row(md2)

  print('Dataframe without Rolling Mean')
  print(x)
  print(w)
  print('\n')
  print('Dataframe with Rolling values')
  print(y)
  print(z)
```

In [14]:

```
def cal_roll(win_size,data):
  feat = ['eeg_fp1', 'eeg_f7', 'eeg_f8', 'eeg_t4', 'eeg_t6', 'eeg_t5','eeg_t3', 'eeg_fp2', 'eeg_o1'
, 'eeg_p3', 'eeg_pz', 'eeg_f3', 'eeg_fz','eeg_f4', 'eeg_c4', 'eeg_p4', 'eeg_poz', 'eeg_c3', 'eeg_cz
', 'eeg_o2','ecg', 'r', 'gsr']
  data2 = pd.DataFrame()
```

```
  for f in feat:
    data2[f+'_roll_mean'] = data[f].rolling(win_size,min_periods=1).mean()
    data2[f+'_roll_median'] = data[f].rolling(win_size,min_periods=1).median()
    data2[f+'_roll_max'] = data[f].rolling(win_size,min_periods=1).max()
    data2[f+'_roll_min'] = data[f].rolling(win_size,min_periods=1).min()
    data2[f+'_roll_var'] = data[f].rolling(win_size,min_periods=1).var()
    data2[f+'_roll_25q'] = data[f].rolling(win_size,min_periods=1).quantile(0.25)
    data2[f+'_roll_75q'] = data[f].rolling(win_size,min_periods=1).quantile(0.75)
    data2[f+'_roll_iqr'] = data2[f+'_roll_75q'] - data2[f+'_roll_25q']

  return data2
```

In [24]:

```
def make_plot(data):
  feat = ['event','ecg_roll_mean', 'r_roll_mean', 'gsr_roll_mean','eeg_fp1_roll_mean','eeg_o2_roll_
mean']
  for f in feat:
    for i in range(0,3):
      get_plot(data,f,i)
```

- Let's take pilot from 3rd crew and and seat 0

In [8]:

```
temp_y = df_tr_ti['event'].loc[3,0]
df_tr_ti.loc[3,0]
```

Out[8]:

	time	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2	eeg_o1	
experiment											
0	0.007812	-5.551450	8.574580	-6.417600	-9.36670	-6.207050	1.809690	3.26585	-6.865370	-1.991140	
0	0.011719	-12.028400	1.447700	-12.046300	-11.54220	-3.796700	16.667101	-5.89147	-13.834700	4.860700	
0	0.015625	-12.876000	-2.018680	-15.167200	-10.97040	-8.148430	3.569330	-4.14559	-15.063900	7.602700	
0	0.019531	-10.158400	0.182365	-12.651200	-9.18370	-13.110600	3.637020	-6.34576	-11.952600	-0.919624	
0	0.023438	-8.103310	2.303100	-10.148500	-7.30021	-4.454200	-5.165750	-5.99792	-10.228600	-14.887300	
...	...	...	...	...	...	...	...	...	...	...	
2	359.910156	-2.525680	-25.352301	29.743700	-1.00924	-6.040700	5.871490	-3.50630	12.296600	1.314820	
2	359.914062	-0.675297	-11.881700	20.063400	1.63667	-0.310786	3.348320	-5.46269	9.548200	8.198560	
2	359.917969	5.467780	2.330660	33.518799	15.16520	8.378880	8.102430	3.13749	17.274500	18.665501	
2	359.921875	9.251840	-12.833400	38.877899	15.49130	11.596700	8.677310	6.41058	21.922001	9.265130	
2	359.925781	6.857010	-7.922790	33.956699	23.26800	9.161260	-4.697920	-12.95890	15.320500	0.353880	

276402 rows × 25 columns

In [22]:

```
temp_y = df_tr_ti.loc[3,0]['event']
temp_t = df_tr_ti.loc[3,0]['time']
temp2 = cal_roll(50,df_tr_ti.loc[3,0])
temp2['event'] = temp_y
temp2['time'] = temp_t
```

```
temp2['time'] = temp_t
temp2
```

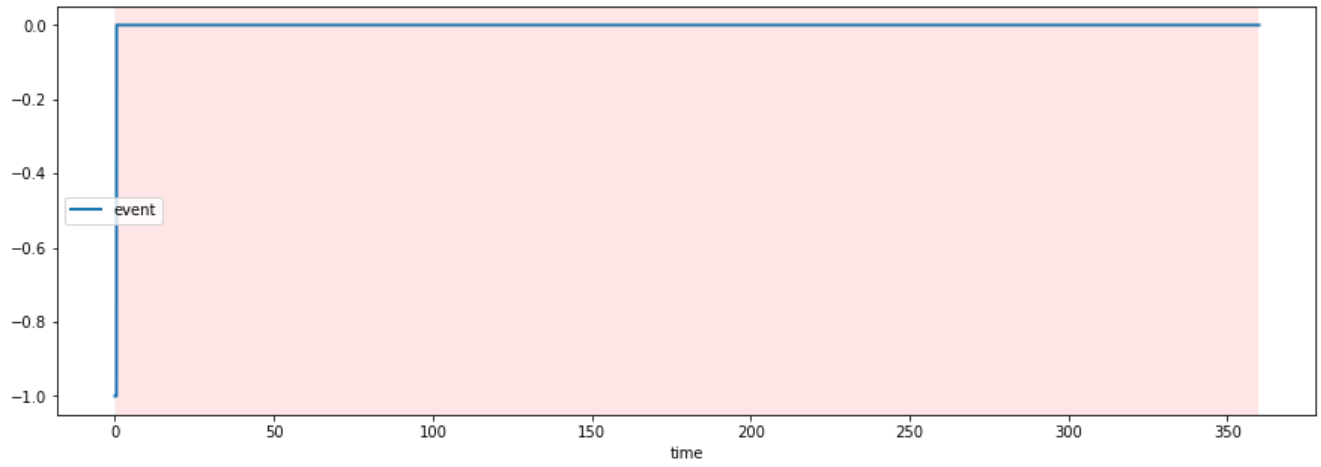
Out[22]:

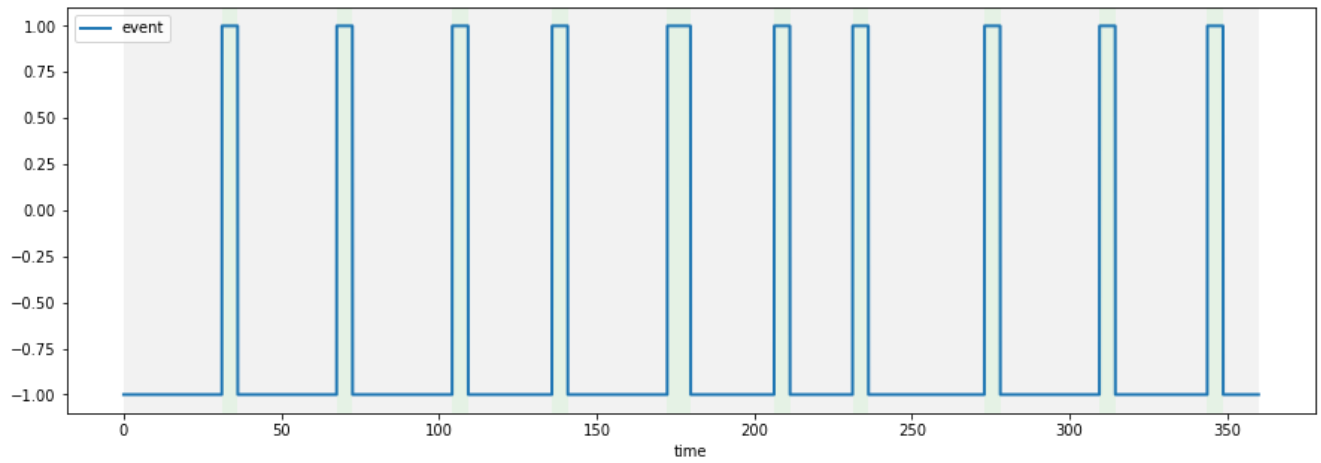
	eeg_fp1_roll_mean	eeg_fp1_roll_median	eeg_fp1_roll_max	eeg_fp1_roll_min	eeg_fp1_roll_var	eeg_fp1_roll_
experiment						
0	-5.551450	-5.551450	-5.55145	-5.55145	NaN	-5.551450
0	-8.789925	-8.789925	-5.55145	-12.02840	20.975441	-10.409162
0	-10.151950	-12.028400	-5.55145	-12.87600	16.053057	-12.452200
0	-10.153562	-11.093400	-5.55145	-12.87600	10.702048	-12.240300
0	-9.743512	-10.158400	-5.55145	-12.87600	8.867243	-12.028400
...	...	...	...	...	...	...
2	2.750985	3.299650	13.81220	-14.04650	38.016278	-0.255008
2	2.703246	3.299650	13.81220	-14.04650	38.231487	-0.279969
2	2.611539	3.299650	13.81220	-14.04650	37.276422	-0.279969
2	2.520332	3.299650	12.99330	-14.04650	35.607501	-0.279969
2	2.511246	3.299650	12.99330	-14.04650	35.522792	-0.279969

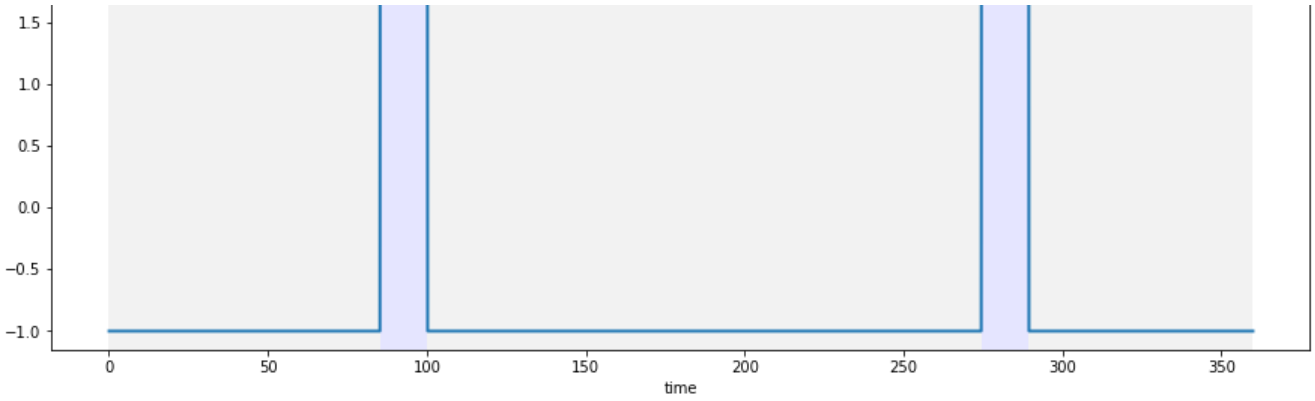
276402 rows × 186 columns

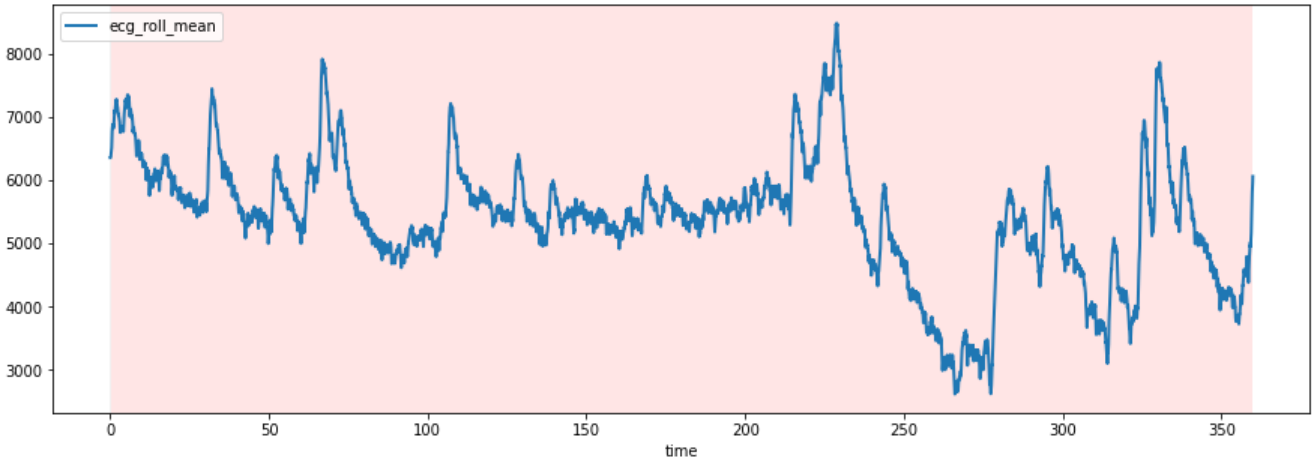
In [25]:

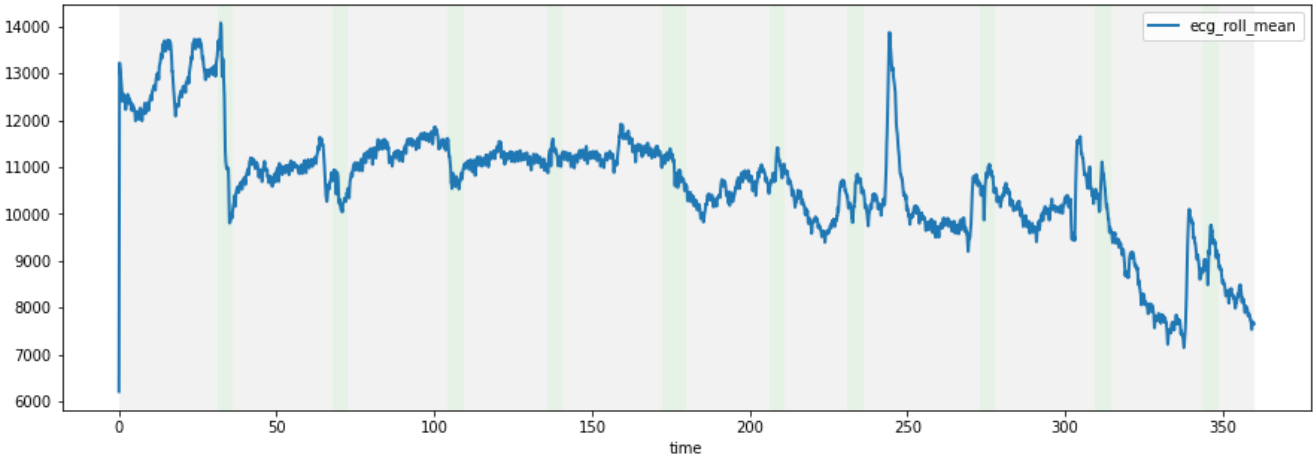
```
make_plot(temp2)
```

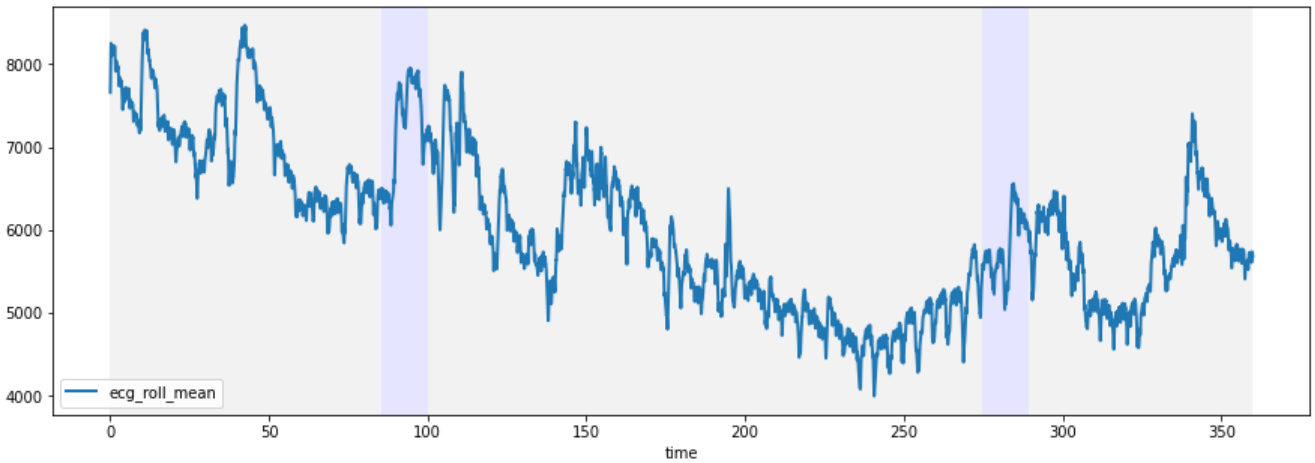


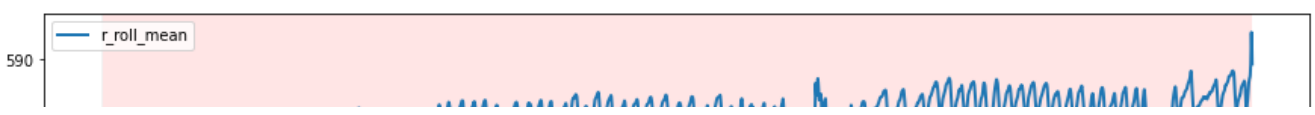


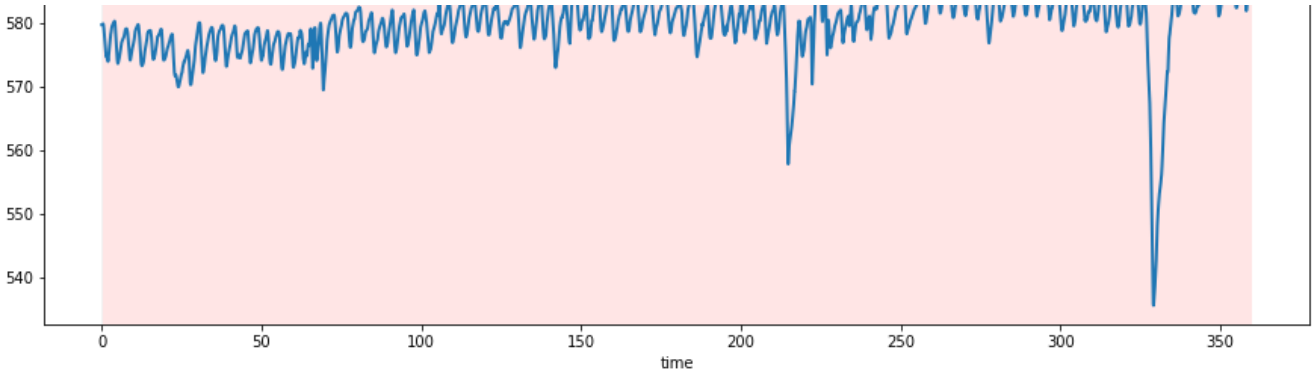

1.5
1.0
0.5
0.0
−0.5
−1.0
0
50
100
150
200
250
300
350
time

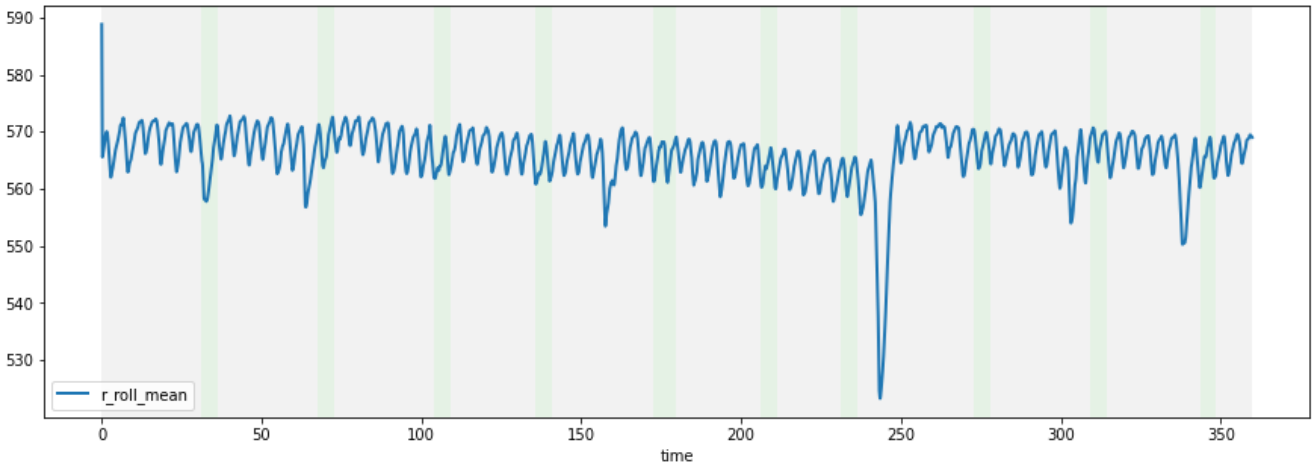

ecg_roll_mean
8000
7000
6000
5000
4000
3000
0
50
100
150
200
250
300
350
time

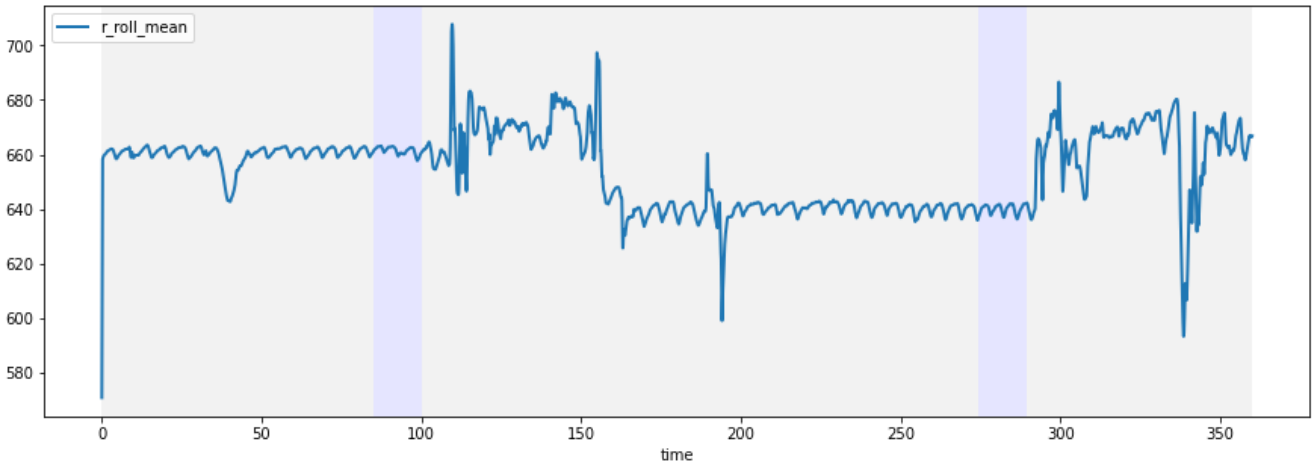

ecg_roll_mean
14000
13000
12000
11000
10000
9000
8000
7000
6000
0
50
100
150
200
250
300
350
time

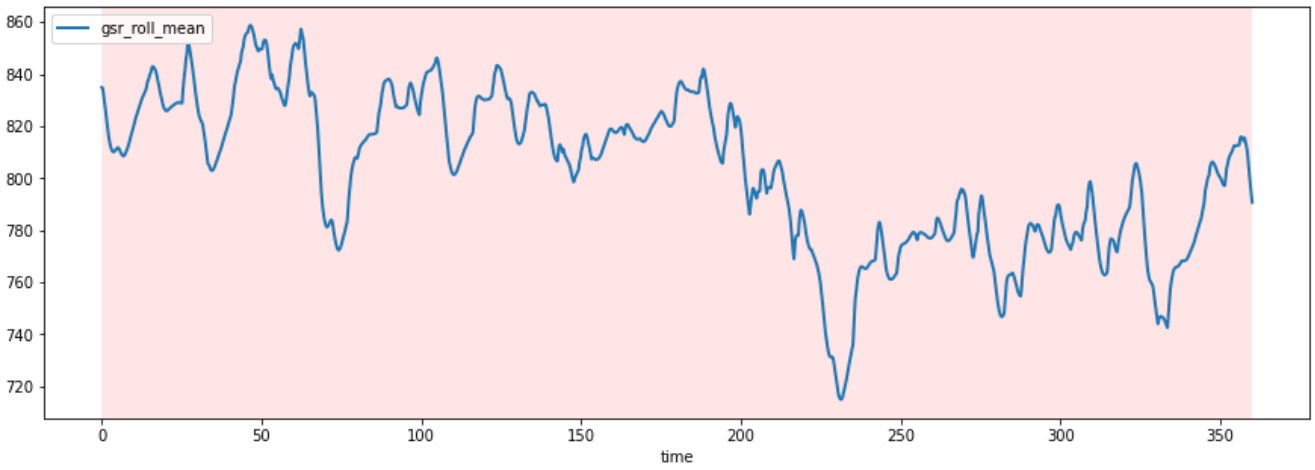

8000
7000
6000
5000
4000
ecg_roll_mean
0
50
100
150
200
250
300
350
time

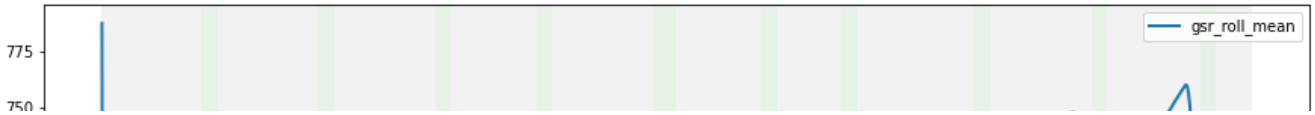

r_roll_mean
590

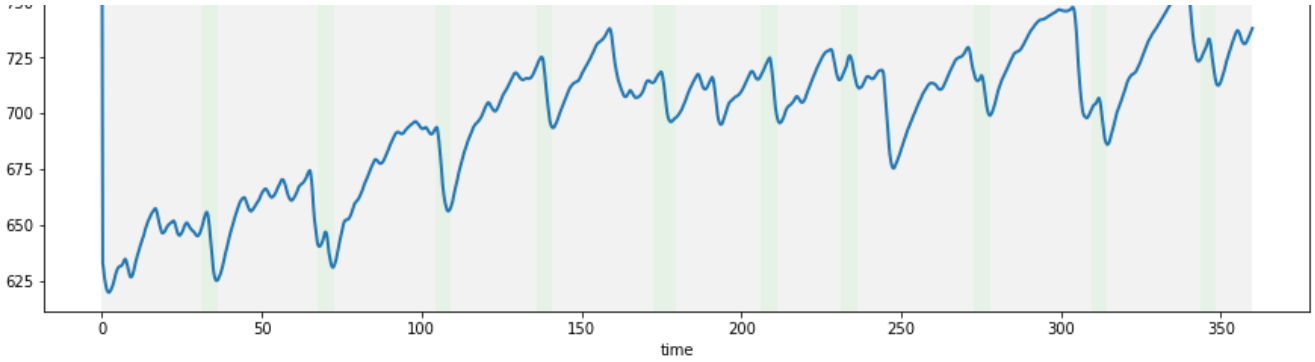

580
570
560
550
540
0
50
100
150
200
250
300
350
time

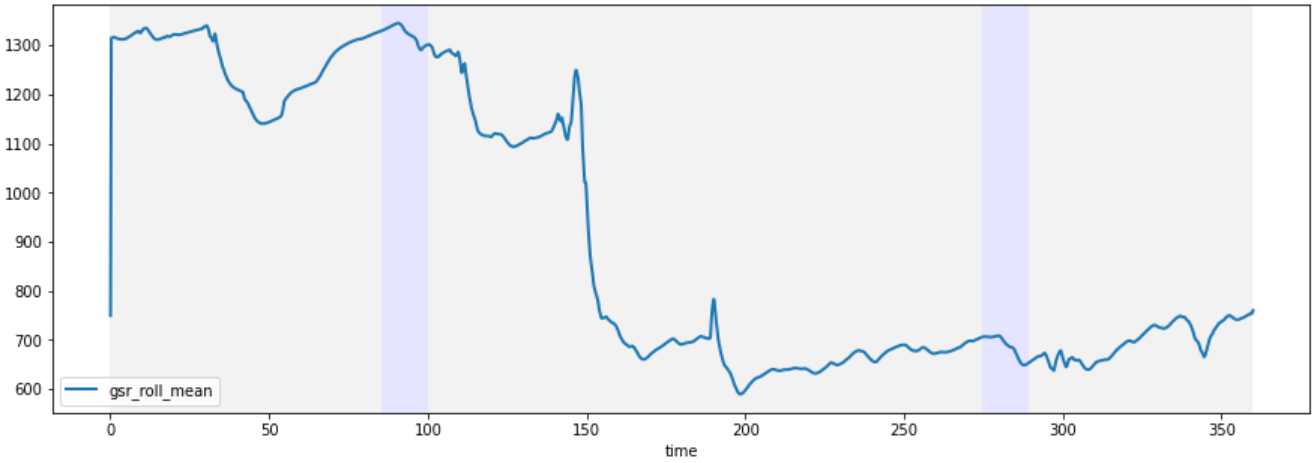

590
580
570
560
550
540
530
r_roll_mean
0
50
100
150
200
250
300
350
time

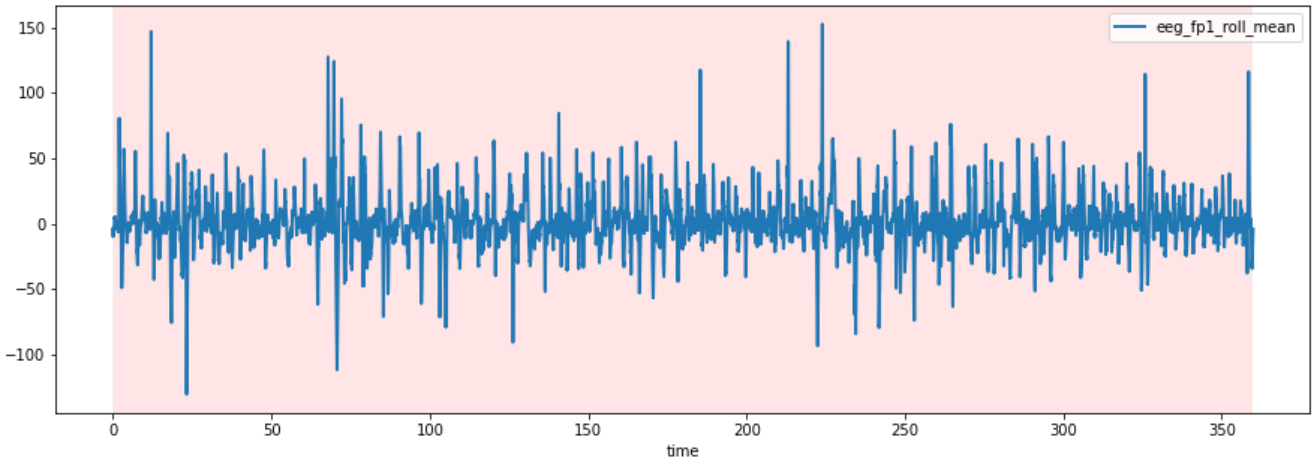

r_roll_mean
700
680
660
640
620
600
580
0
50
100
150
200
250
300
350
time

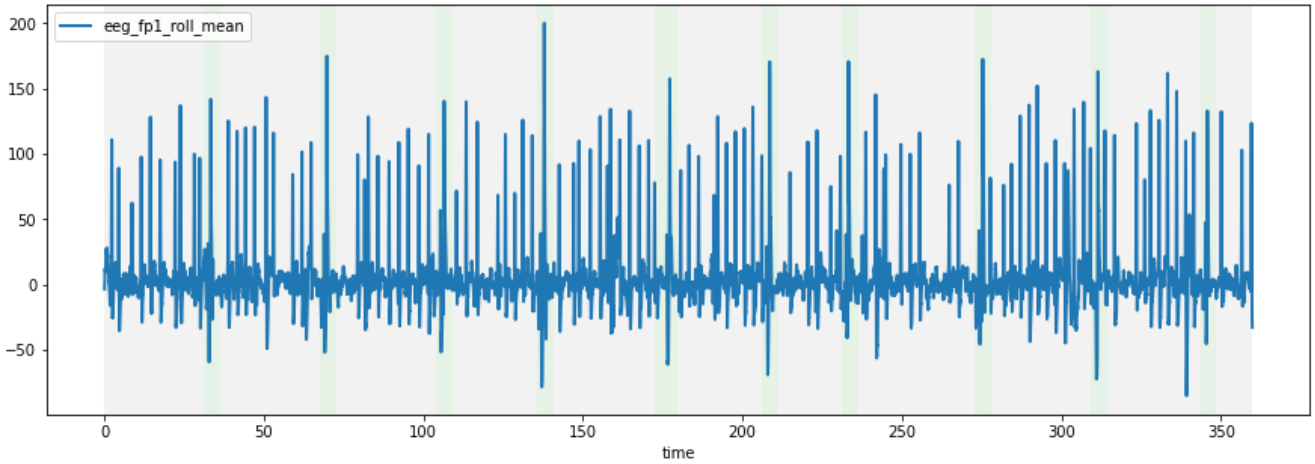

gsr_roll_mean
860
840
820
800
780
760
740
720
0
50
100
150
200
250
300
350
time

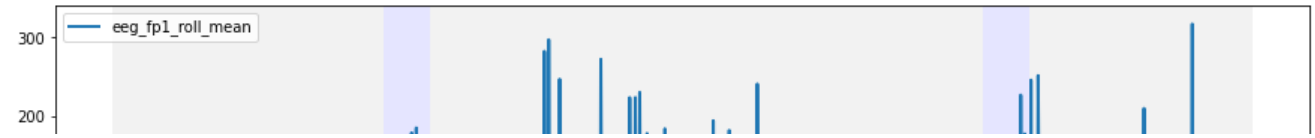

gsr_roll_mean
775

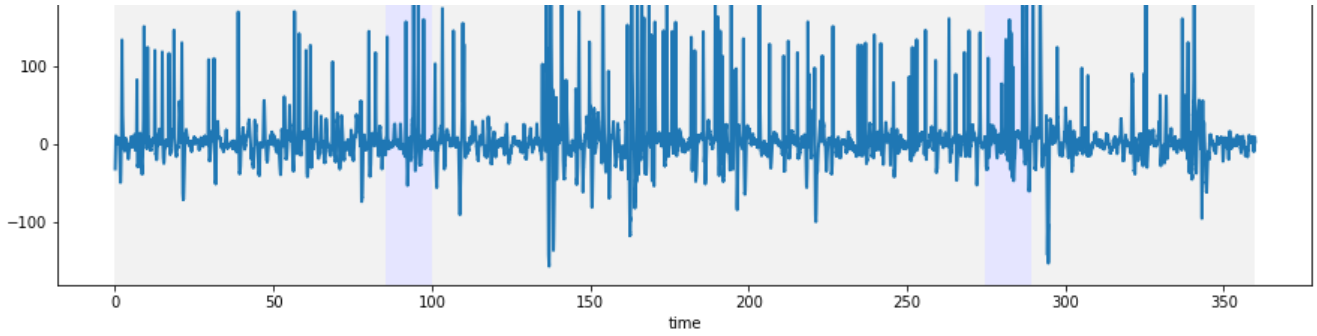

725
700
675
650
625
0
50
100
150
200
250
300
350
time

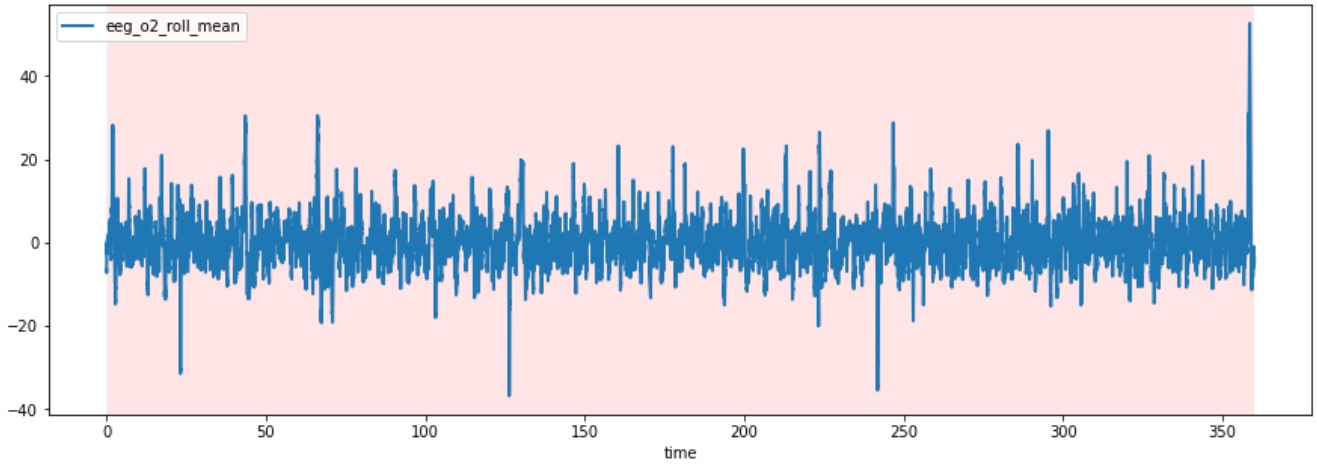

1300
1200
1100
1000
900
800
700
600
gsr_roll_mean
0
50
100
150
200
250
300
350
time

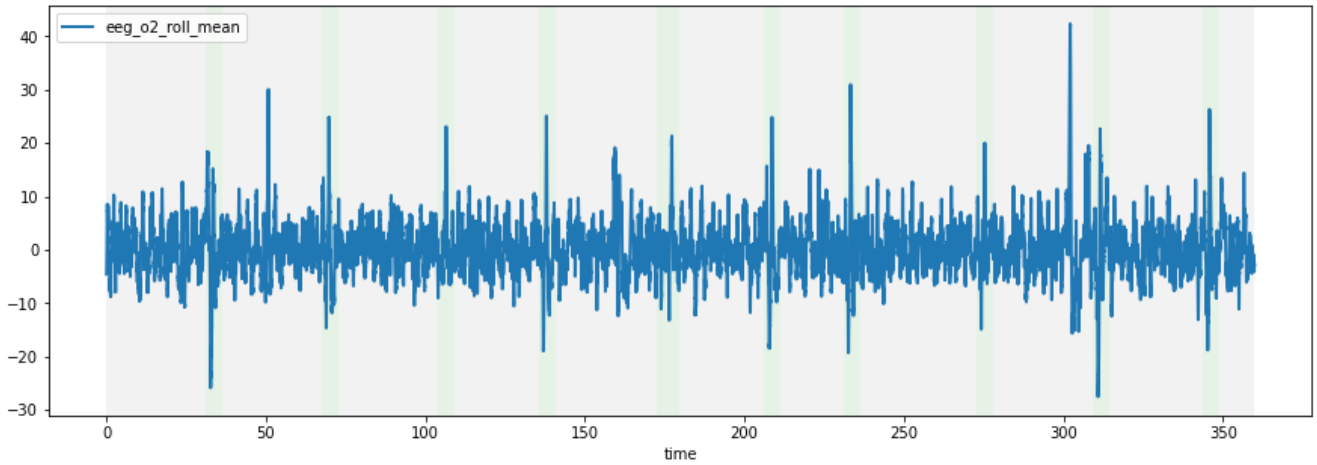

150
100
50
0
−50
−100
eeg_fp1_roll_mean
0
50
100
150
200
250
300
350
time

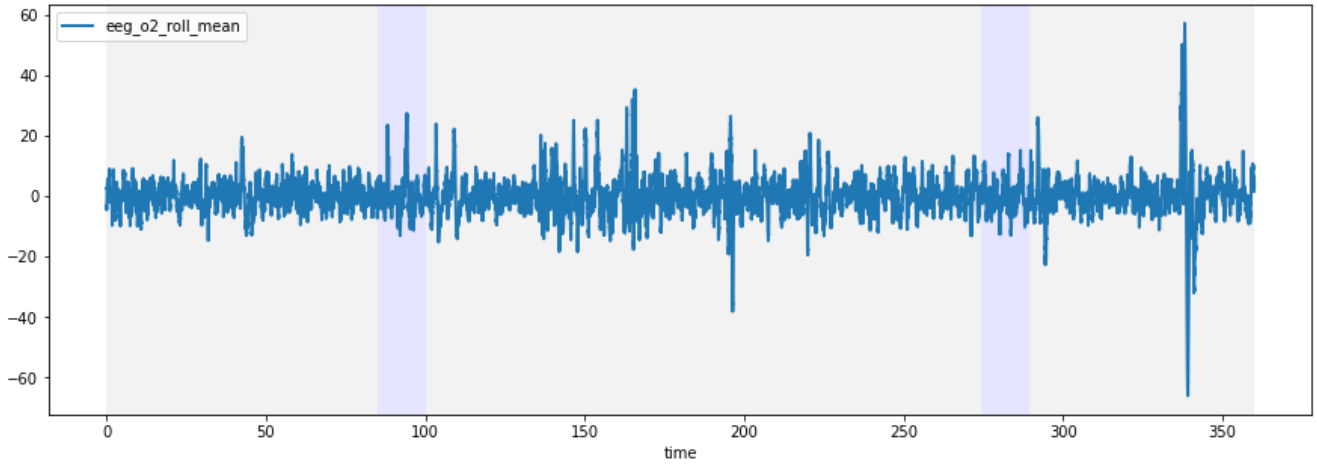

200
150
100
50
0
−50
eeg_fp1_roll_mean
0
50
100
150
200
250
300
350
time


300
200
eeg_fp1_roll_mean









In [40]:

```
calculate_values(df_tr_ti.loc[3,0],temp2)
```

```
Dataframe without Rolling Mean
```

```
+-----------------+--------------------+--------------------+--------------------+-------------
----+---------------------+----------------------+--------------------+---------------------+---
---------------+--------------------+--------------------+---------------------+---------------
-+---------------------+----------------------+--------------------+--------------------+-----
-------------+--------------------+-------------------+-------------------+--------------------+
-----------------+
|      Value      |       eeg_fp1       |        eeg_f7        |        eeg_f8        |        eeg_t4
|       eeg_t6        |         eeg_t5         |        eeg_t3        |       eeg_fp2        |
eg_o1       |        eeg_p3        |        eeg_pz        |        eeg_f3        |       eeg_fz
|       eeg_f4        |         eeg_c4         |        eeg_p4        |       eeg_poz        |
eeg_c3       |        eeg_cz        |      eeg_o2       |        ecg        |         r         |
gsr         |
+-----------------+--------------------+--------------------+--------------------+-------------
----+---------------------+----------------------+--------------------+---------------------+---
---------------+--------------------+--------------------+---------------------+---------------
-+---------------------+----------------------+--------------------+--------------------+-----
-------------+--------------------+-------------------+-------------------+--------------------+
-----------------+
|       Mean      |  4.641538724043375  |  1.2226086339208813 |  1.9543162202154714 |
0.17721251749625053 | -0.09951015177893296 | -0.017298428499072198 | 0.15526453091873554 |
4.1126560520256294 | 0.11063873349324402 | 0.17489356344744514 | 0.24547718407971442  |
1.353157565079126   | 1.5210187843033043 | -0.22619459463028585 |   0.7714821220215583  |
0.2864545668917004  | 0.22409908807823328 |  0.5514156343152444 | 0.48391208756449294 |
0.197075452142164  | 7377.148721716116  | 600.1418350108189  |  805.508498751251  |
|      Median     | 0.40779750000000003 | 0.05756850000000001 | 0.15040900000000001 |         0.0
|        0.0          |       -0.0105135       |         0.0          | 0.39682649999999997 |       -
.041392      |      -0.0589365      | -0.11845649999999999 | 0.057730500000000004 |      0.0089745
|       0.141291       | -0.015869499999999998 | -0.08624100000000001 |       -0.0768625      |
0.0         |      -0.047433       |      -0.0388245      |     6350.899902      |     580.5214845     |
742.2579955      |
|     Variance    |  1366.9350851683519 |   556.378842611354  |  1104.8406147845708 |  353.71642743
845604 |  113.14657881839085  |   235.4482834438534   |  939.7462404217962  |  1414.5254663785213 |
148.52013916414174 |  135.77514469377132 |  158.56020589470546  |  453.61793997824896  | 365.873759
09682476 |   7999.511823065811   |    407.08701948901216   |  165.29130385418344  |
130.84928076055218 |  262.9216662259213  |  194.75638966613093 | 131.59687824373873 |
6608275.3805270605 | 1593.8879930735989 | 34780.39627640001  |
|       Max       |  440.94000199999994 |       366.92099       |       480.329987      |       352.6210
2     |      121.533997      |       382.542999      |       792.008972       |      447.618988     |
119.779999      |       206.466995      |  183.59399399999998  |  654.7069700000002   |
265.60598799999997 |  2034.1700440000002  |          692.75          |       319.432007      |       158
128006       |       506.221008       |       212.231003      |       141.707001      |      14482.900391     |
708.051025      |      1345.219971      |
|       Min       |        -211.459       |        -400.944       |      -420.368988      |       -
309.79599      |      -130.279007      |  -365.04699700000003  |      -416.825012      |       -
225.843002      |      -152.651993      |      -169.389999      | -151.83500700000005  | -
453.88598600000006  |      -171.940994      |      -2333.830078      |  -481.26098600000006  | -255.4709
9300000002  |      -151.662994      | -452.55999800000006 |      -144.186005      |      -131.104004
|     2337.310059     |     523.094971     |     590.10199      |
| 25th percentile |       -9.6326225      |       -8.4593425      |       -10.005675      |       -
8.711215       |  -6.151649999999999  |        -6.99149        |       -8.90746       |       -9.289107
|  -7.263054999999999 |  -6.849445000000001  |  -7.456840000000001  |  -7.821160000000001  | -7.637
222500000001 |      -16.64497475      |   -7.519777499999999  |       -6.3812125       |  -
6.882507500000002 |  -7.131287499999999 |        -7.60999       |       -6.90176       |      5395.755005
|      568.896973      |     696.01925625     |
| 75th percentile |  10.633275000000001 |        8.98344        |        10.7794        |       8.761512
|       6.0614975       |         6.8743675        |  8.644110000000001  |  10.229624999999999 |
7.290347499999999  |  6.847955000000001  |  7.430605000000003  |  8.284302500000003  | 8.0797900C
0000001  |  17.008149749999998  |   7.783302500000001   |       6.4067575       | 6.938750000000001
|  7.313902500000001  |        7.74634       |       6.9784875       |      10063.700195     |       641.317
93     | 819.7520139999997  |
|       IQR       |       20.2658975      |       17.4427825      |       20.785075       |
17.472727499999998 |  12.213147499999998  |        13.8658575       |        17.55157       |       19.5
87325      |  14.553402499999997 |  13.697400000000002  |  14.887445000000003  |       16.1054625
| 15.717012500000003 |       33.6531245       |   15.303080000000001  |  12.787970000000001  |  13.82
1257500000002 |        14.44519        |        15.35633        |       13.8802475       | 4667.945189999999
|       72.42102       | 123.73275774999968 |
+-----------------+--------------------+--------------------+--------------------+-------------
----+---------------------+----------------------+--------------------+---------------------+---
---------------+--------------------+--------------------+---------------------+---------------
-+---------------------+----------------------+--------------------+--------------------+-----
-------------+--------------------+-------------------+-------------------+--------------------+
-----------------+
+-------+-------------+
| Value |    event    |
+-------+-------------+
|  Mode |    0   -1   |
|       | dtype: int8 |
```

```
          --
+-------+-------------+


Dataframe with Rolling values
+-----------------+---------------------+---------------------+---------------------+---------------
-----+---------------------+---------------------+---------------------+---------------------+--
---------------+---------------------+---------------------+---------------------+---------------------+-----
-+---------------------+---------------------+---------------------+---------------------+-----
-------------+---------------------+---------------------+---------------------+-----------------
+--------------------+
|      Value      |       eeg_fp1       |        eeg_f7       |        eeg_f8       |        eeg_t4
|        eeg_t6       |         eeg_t5        |        eeg_t3       |       eeg_fp2       |
eg_o1       |        eeg_p3        |        eeg_pz       |       eeg_f3       |        eeg_fz
eeg_f4        |        eeg_c4        |        eeg_p4        |       eeg_poz        |        eeg_c3
|        eeg_cz       |        eeg_o2        |        ecg         |         r          |        gsr
|
+-----------------+---------------------+---------------------+---------------------+---------------
-----+---------------------+---------------------+---------------------+---------------------+--
---------------+---------------------+---------------------+---------------------+---------------------+-----
-+---------------------+---------------------+---------------------+---------------------+-----
-------------+---------------------+---------------------+---------------------+-----------------
+--------------------+
|       Mean      |  4.640814643391568  |  1.2231545735335685 |  1.951406001497336  |
0.17597007434261927  | -0.10015680040286623 | -0.017453076219987797 |  0.1553493608103128 |
4.111017160723313  | 0.10935087386733916 | 0.17370859240966247  | 0.24320704775015026 |
1.3526862249438822 |  1.5197890330921544 | -0.23437223749892683 |  0.7697374416787589  |
0.2853473725304884  |  0.2231769608635823  |  0.5503869214865538  |  0.4823405075987455 |  0.196217
4167945201  | 7377.209212697445  | 600.1341082528444  | 805.5150655671972  |
|      Median     |  0.8085674799999758 | 0.17740238000032466 | 0.19397398000056343 | -0.0687421900
0002882 | -0.2476733900000106  |  -0.22042078000013338 | -0.5365202300013692 |  0.7538588600006042
| -0.1680503000000819 | -0.21168200999980363 | -0.2531581899999489 | 0.2314013399998301 |
0.10754234000006915 |  0.3647601300003124  | 0.016568239999875643 | -0.20627469000021526 | -
0.18491017000007906 | -0.09966790000011788 | -0.1754690599998342 | -0.04068623000007825 |
6350.7690916501515 | 580.5245635700004  | 742.2837945700178  |
|     Variance    |   896.252952463469  |  362.50416063114864 |  799.9803078099527  |  165.52973440
734516  |  34.74451110249825   |   94.72666890912011   |   694.246116075435  |  985.4187780033811
|  35.01995975755767  |  49.30526307239286   |  78.34820927796859  | 223.7163650625864  |  229.2026
948499444  |  6719.171262724338   |  143.05339123333943  |  55.86811905487043   |   51.472621387059
2   |  104.26805917088892  |  103.41852379307116 |  35.70106592773614   | 6600526.429979919  | 1593
.0042125444131 | 34769.12197582644  |
|       Max       |  317.5340417200002  |  262.97889976000124 |  369.6149008400012  |
143.1568122799999   |  60.834489879999964  |   165.18139590000007  |  676.4918963999994  |  296.894
3386600007  |  44.66324977999986  |  61.936056320000425  |  57.34898801999994  | 164.07819941999955
|  129.9052792799998  |  1427.3791553000008  |  131.0231877600001   |  81.95478745999982   |  42.23
260811999977   |  123.51732449999984  |  65.48546526000023  |  57.41957269999986   | 14076.734120980
205 | 707.7817809000184  | 1345.1986012600128 |
|       Min       | -156.86748011999993 | -231.27820037999874 |  -400.1049810799988 |  -127.9432255
400001  |  -61.67040374000007  |   -81.4993699800002   | -220.29031924000003 | -182.88193997999954
|  -42.14733996000011 |  -57.36235413999998  |  -67.82290212000007 | -105.2346930000002 |  -91.972
45014000003 | -2006.3978100199986  | -103.16268613999999  | -55.611303040000095  |  -
53.38619000000006  |  -88.48419378000023  |      -71.758674     |  -66.04171786000018  | 2621.21858
8980115  | 523.3155944200083  | 590.1133860600145  |
| 25th percentile |  -7.147749029999863 |  -5.562613724999709 |  -6.430465899999215 |  -
3.96470472000003   |  -2.745727905000088  |   -2.906491280000063  |  -3.913032585001226 |  -
6.888929394999309 | -3.4566040850000284 |  -3.921825049999638  |  -5.045244190000021 | -
5.286118710000207 | -5.1256447500001165 | -15.181413424999649  |  -4.462262680000059  | -
3.6389944850001257  |  -4.175305425000057  |  -4.431201900000021  |  -5.05998056999986  |  -
3.205012390000128  | 5400.234943860127  | 568.8927993600028  |  696.019845555018  |
| 75th percentile |  8.998594675000135  |  6.247547615000297  |  7.665683415000774  |  3.897967699
99997   |  2.393413545000038   |   2.610030854999912   |  3.1709757749985763 |  8.822510580000348
|  3.312629839999852  |  3.7628445650001208  |  4.821689005000024  | 6.026980924999999  |  5.721011
069999937  |  15.130943185000827  |  4.868397920000144   |  3.5417302349997133  |  4.04012787999985
35  |   4.58017558499981   |  5.043176840000211  |  3.0989593499999057  | 10065.130952255211 | 641.
3138852150124  |  819.829760395001  |
|       IQR       |     16.146343705    |  11.810161340000006 |  14.096149314999987 |
7.8626724200000275  |  5.139141450000126   |   5.516522134999975   |  7.084008359999802  |  15.7114
39974999656 |  6.769233924999881  |  7.684669614999759   |  9.866933195000044  | 11.313099635000206
|  10.846655820000054 |  30.312356610000478  |  9.330660600000204   |  7.180724719999839   |   8.21
543330499991   |   9.01137748499983   |  10.10315741000007  |  6.3039717400000335  | 4664.896008395
084  | 72.42108585500966  | 123.80991483998298 |
+-----------------+---------------------+---------------------+---------------------+---------------
-----+---------------------+---------------------+---------------------+---------------------+--
---------------+---------------------+---------------------+---------------------+---------------------+-----
-+---------------------+---------------------+---------------------+---------------------+-----
-------------+---------------------+---------------------+---------------------+-----------------
+--------------------+
+-------+-------------+
```

```
| Value |    event    |
+-------+-------------+
|  Mode |    0   -1   |
|       | dtype: int8 |
+-------+-------------+
```

Let's look at data from 2nd crew seat 1

In [41]:

```
df_tr_ti.loc[2,1]
```

Out[41]:

	time	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2	eeg_
experiment										
0	0.078125	28.280800	9.833440	14.564000	4.374090	4.587000	10.485700	4.986250	32.792999	4.2531
0	0.082031	27.272499	8.141770	17.580299	11.505800	-5.647250	2.702710	0.221040	38.980701	-4.4941
0	0.085938	27.375099	9.296160	17.186701	13.656700	-1.235740	4.507190	-0.583664	35.071499	3.9093
0	0.089844	27.856199	12.497900	15.346700	12.644400	27.334999	6.323910	3.061650	34.418301	6.4801
0	0.093750	27.471001	14.382600	14.605000	9.492630	8.696860	4.422900	0.551582	33.853699	-7.3482
...	...	...	...	...	...	...	...	...	...	...
2	360.203125	-25.235001	11.991600	-87.080200	-49.431999	-46.887402	9.144230	-1.005970	-92.055496	-22.542
2	360.207031	-14.598900	27.528799	-72.768898	-29.729200	-21.973000	28.153700	2.163830	-69.563797	6.0057
2	360.210938	-13.099700	26.093599	-76.024300	-34.820702	-20.823900	29.319401	11.218400	-73.583099	0.7636
2	360.214844	-35.485699	0.000000	-95.374496	-49.406601	-57.511700	14.725900	-7.628150	-90.964996	-31.678
2	360.218750	-50.310501	-15.620100	-103.629997	-62.063099	-62.155201	9.325650	-27.860300	-102.492996	-38.276

276410 rows × 25 columns

In [42]:

```
temp_y2 = df_tr_ti.loc[2,1]['event']
temp_t2 = df_tr_ti.loc[2,1]['time']
temp3 = cal_roll(50,df_tr_ti.loc[2,1])
temp3['event'] = temp_y2
temp3['time'] = temp_t2
temp3
```

Out[42]:

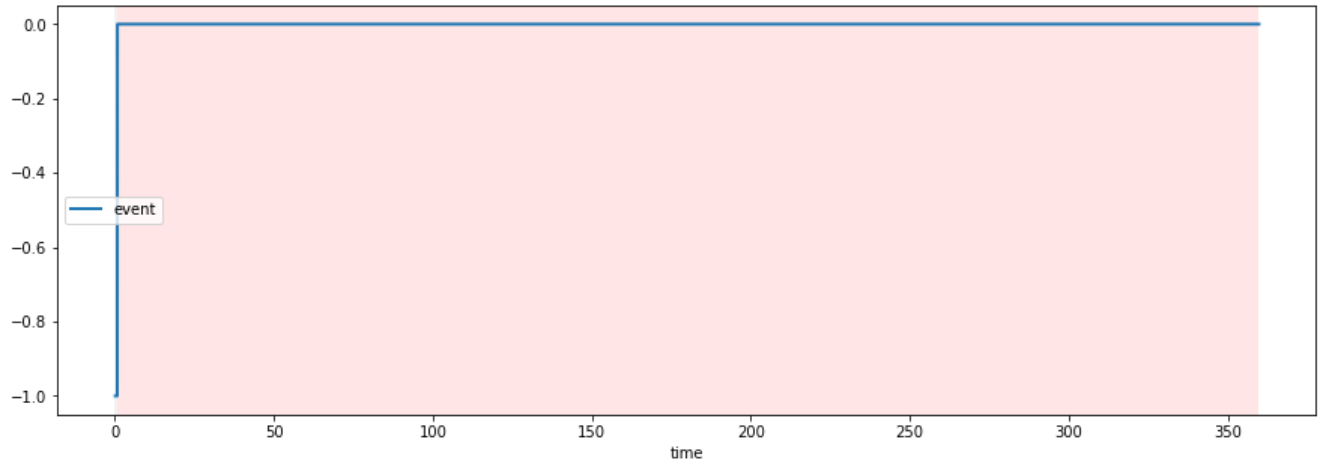
	eeg_fp1_roll_mean	eeg_fp1_roll_median	eeg_fp1_roll_max	eeg_fp1_roll_min	eeg_fp1_roll_var	eeg_fp1_roll_
experiment						
0	28.280800	28.280800	28.280800	28.280800	NaN	28.280800
0	27.776649	27.776649	28.280800	27.272499	0.508335	27.524574
0	27.642799	27.375099	28.280800	27.272499	0.307915	27.323799
0	27.696149	27.615649	28.280800	27.272499	0.216662	27.349449
0	27.651120	27.471001	28.280800	27.272499	0.172635	27.375099
...	...	...	...	...	...	...
2	-44.296228	-45.300500	-21.860399	-74.690804	120.854914	-51.922825

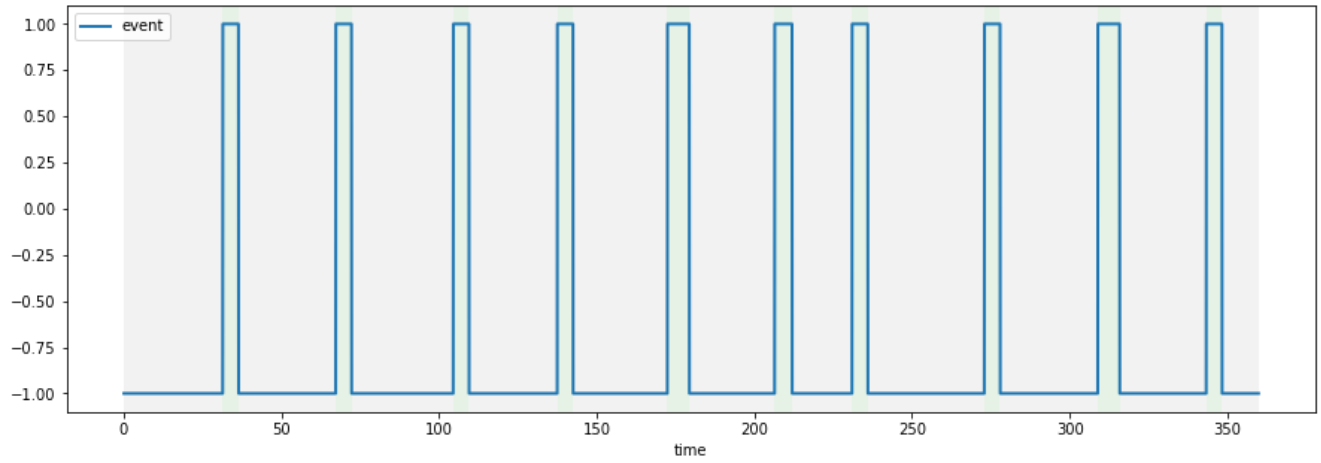
2	-43.094390	-45.291851	-14.598900	-64.403900	118.525941	-50.035599
eeg_fp1_roll_mean	**eeg_fp1_roll_median**	**eeg_fp1_roll_max**	**eeg_fp1_roll_min**	**eeg_fp1_roll_var**	**eeg_fp1_roll_**	
2	-42.068306	-44.619601	-13.099700	-62.003601	126.545201	-49.450623
experiment						
2	-41.608144	-43.739099	-13.099700	-62.003601	121.707372	-49.182199
2	-41.456970	-43.739099	-13.099700	-62.003601	117.833216	-49.182199

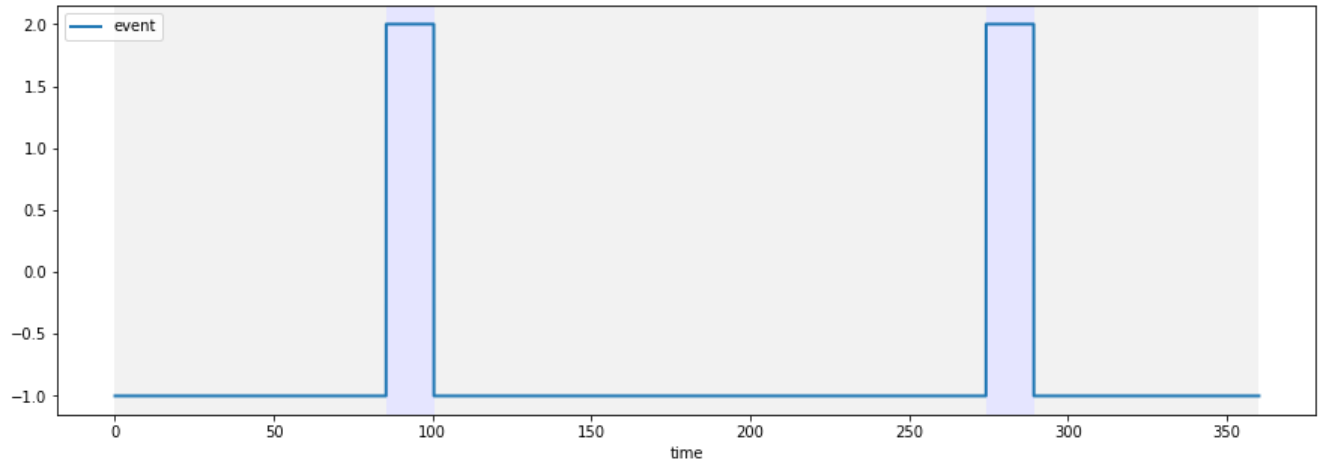
276410 rows × 186 columns

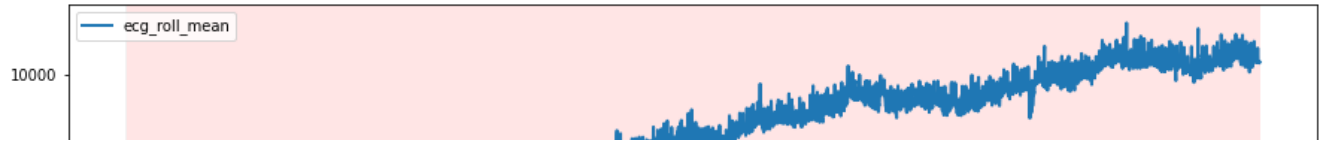
In [43]:

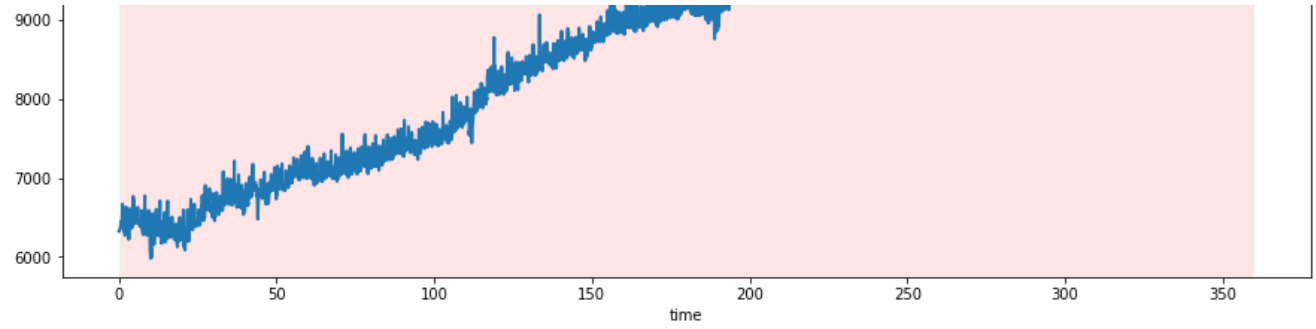
```
make_plot(temp3)
```

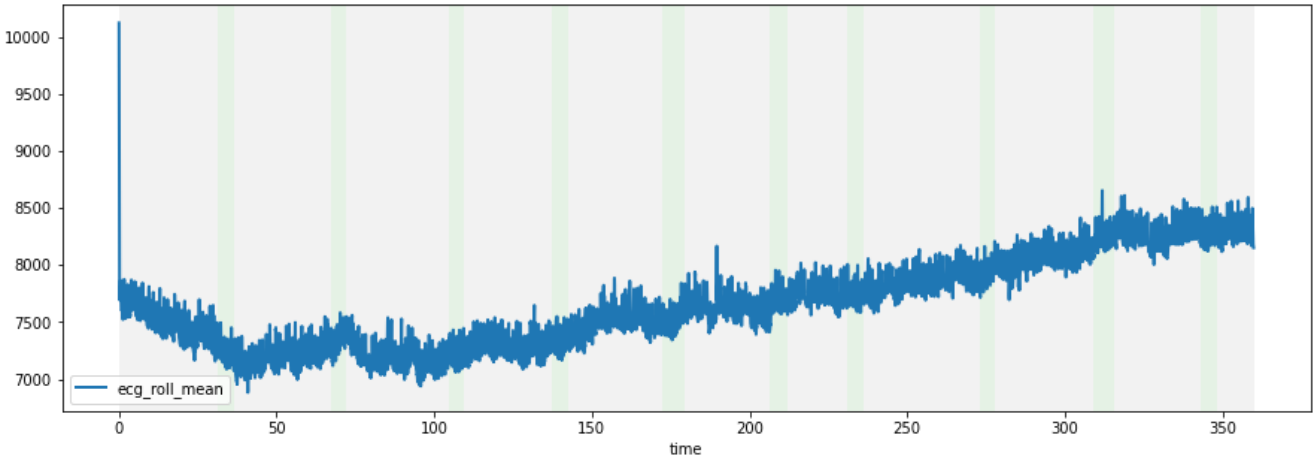


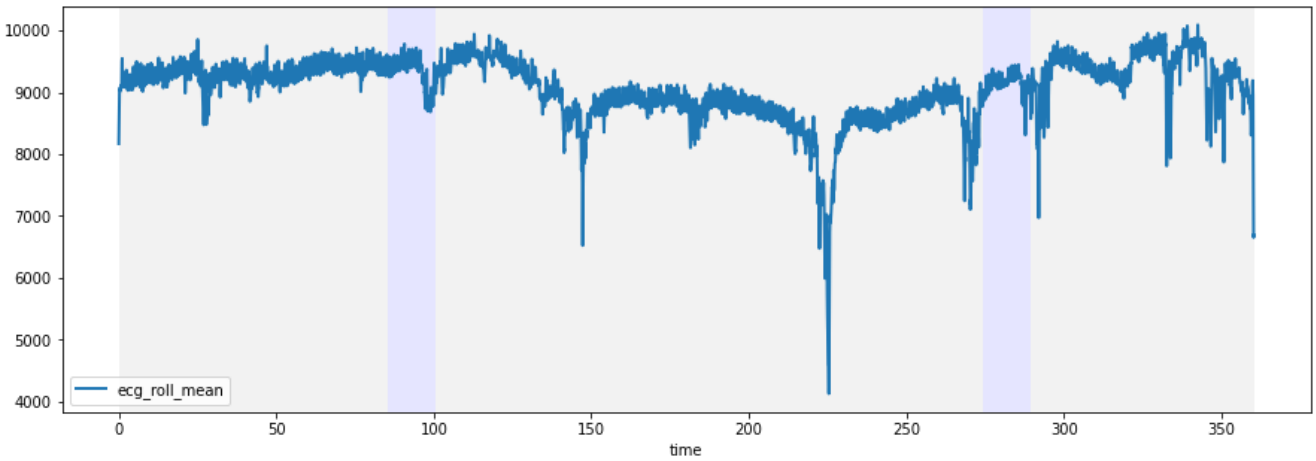


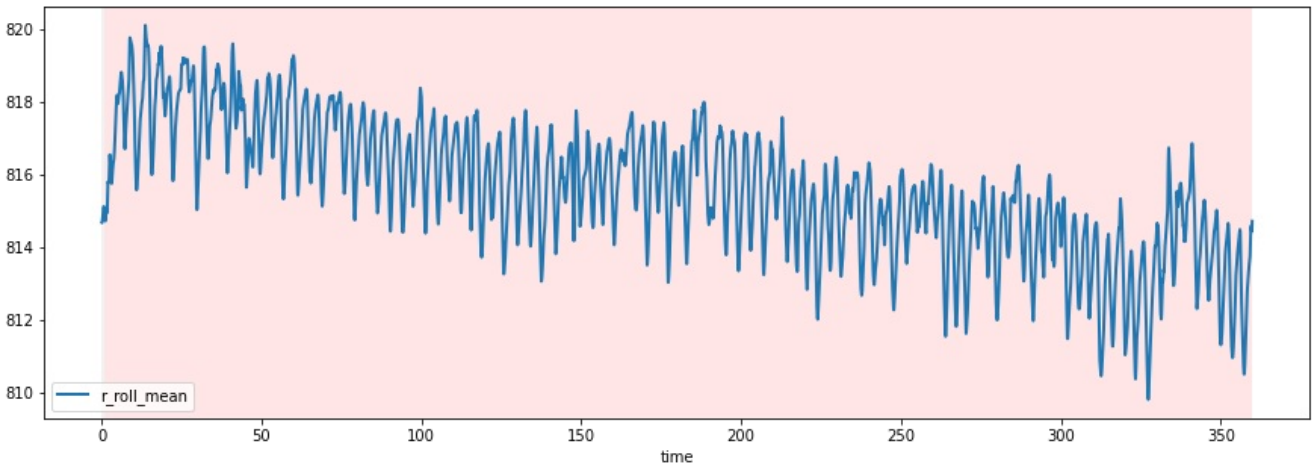


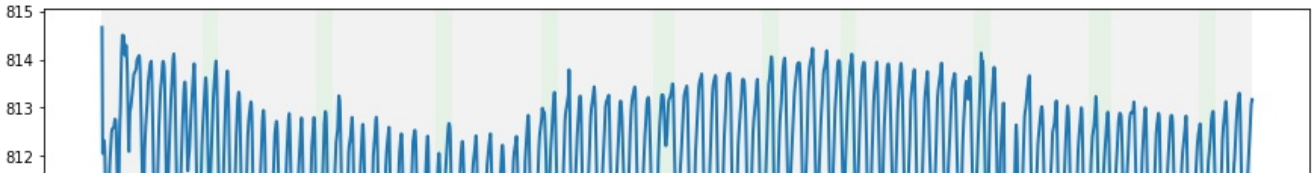


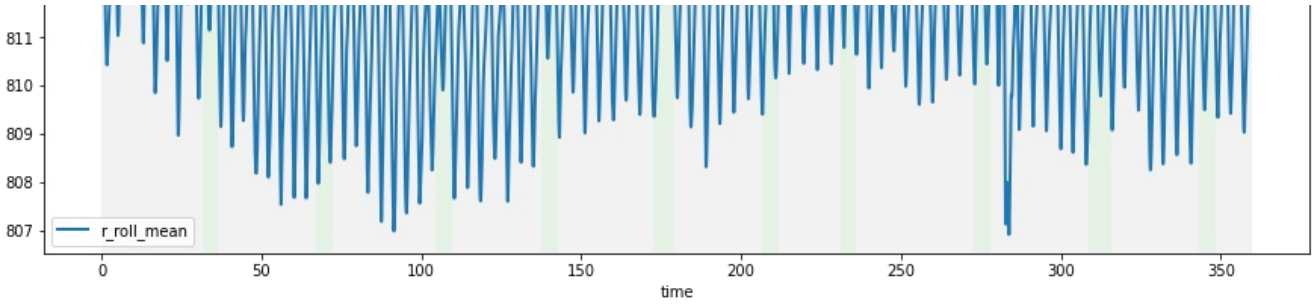

9000
8000
7000
6000
0
50
100
150
200
250
300
350
time

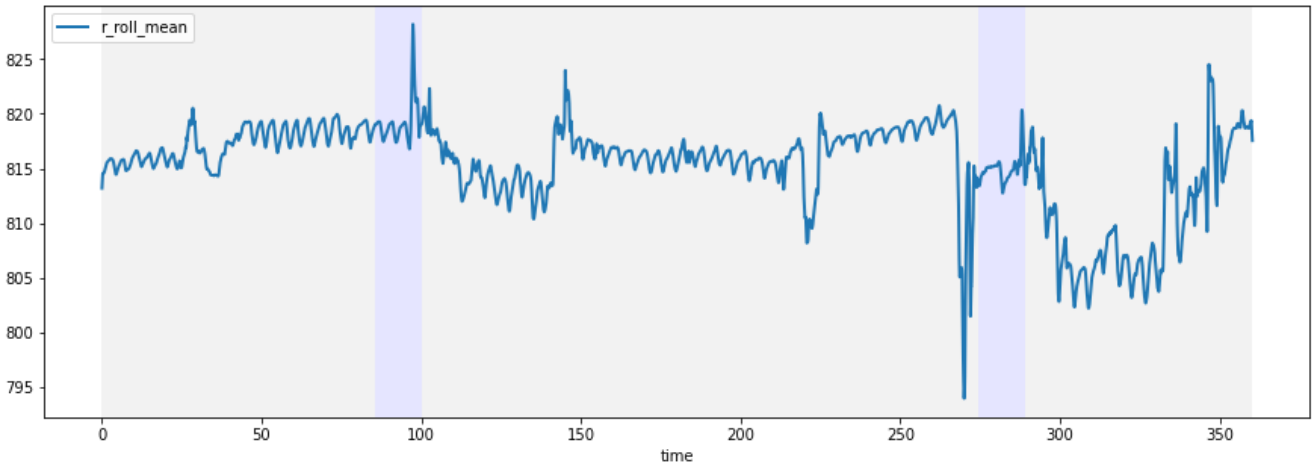

10000
9500
9000
8500
8000
7500
7000
ecg_roll_mean
0
50
100
150
200
250
300
350
time

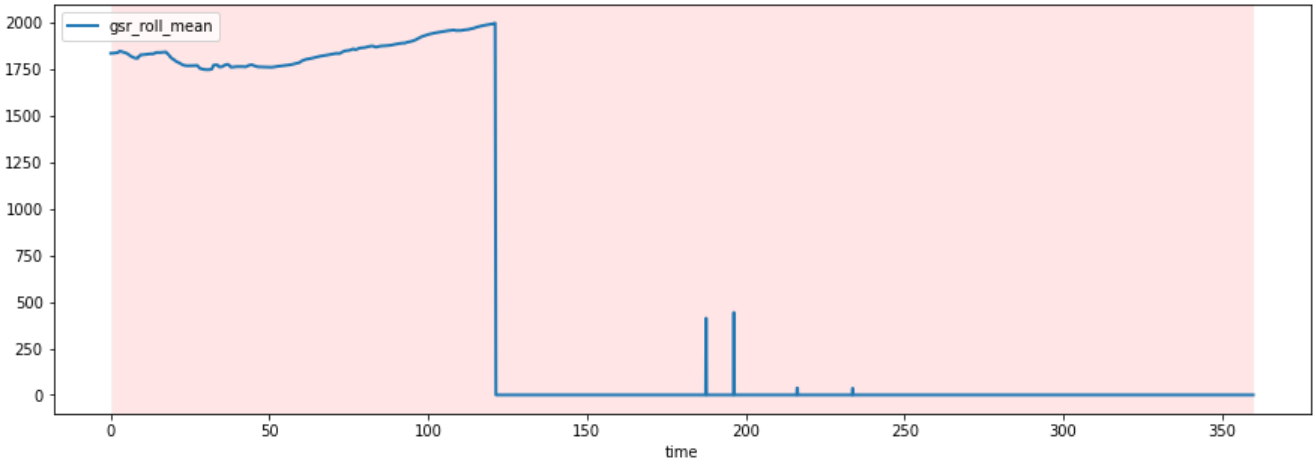

10000
9000
8000
7000
6000
5000
4000
ecg_roll_mean
0
50
100
150
200
250
300
350
time

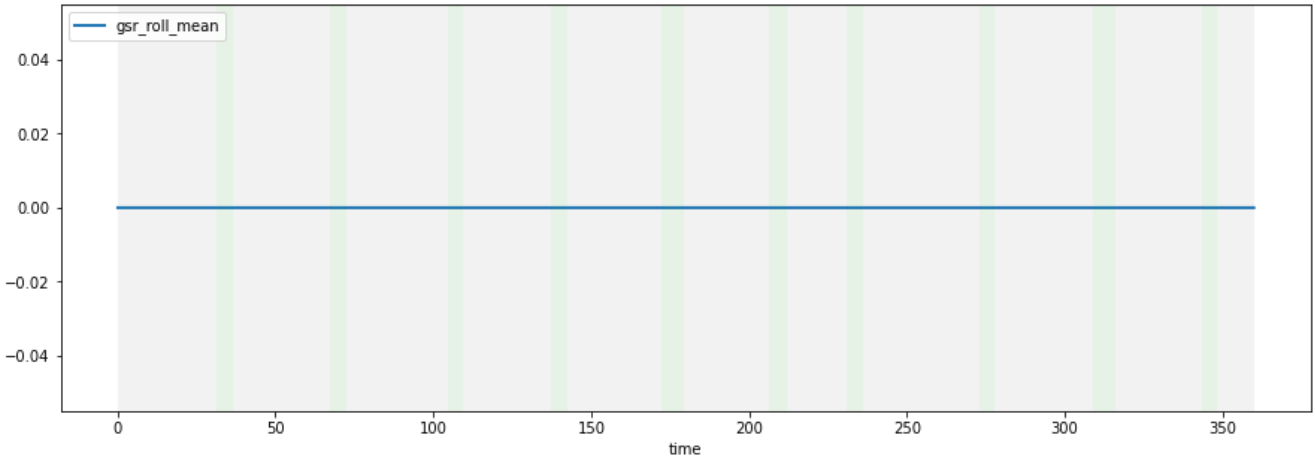

820
818
816
814
812
810
r_roll_mean
0
50
100
150
200
250
300
350
time


815
814
813
812

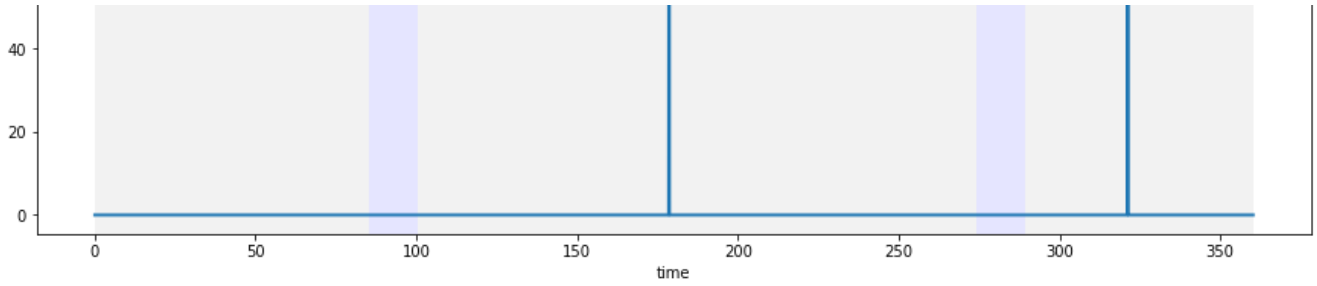

r_roll_mean
811
810
809
808
807
0
50
100
150
200
250
300
350
time

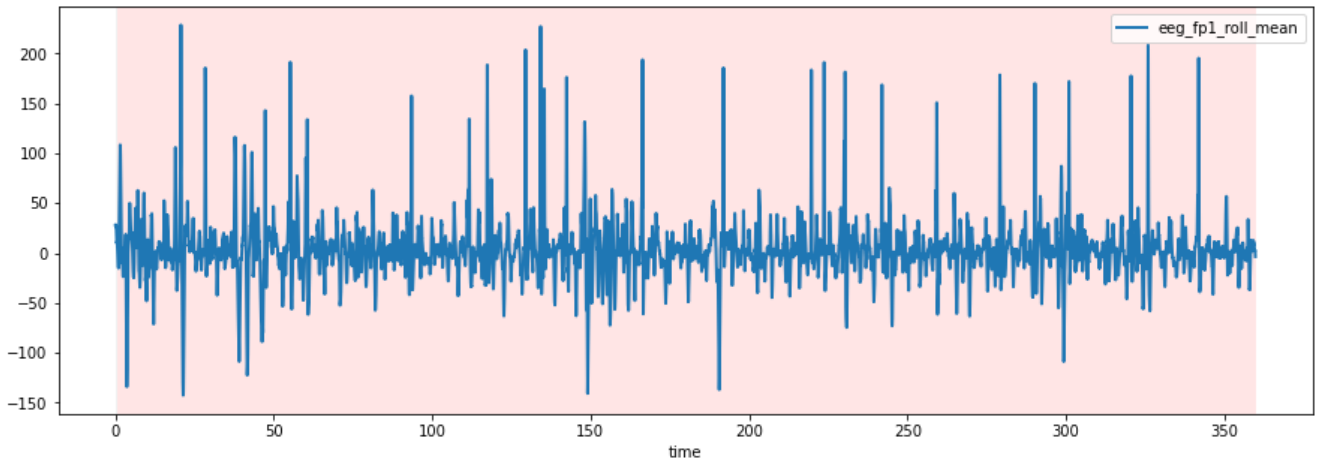

r_roll_mean
825
820
815
810
805
800
795
0
50
100
150
200
250
300
350
time

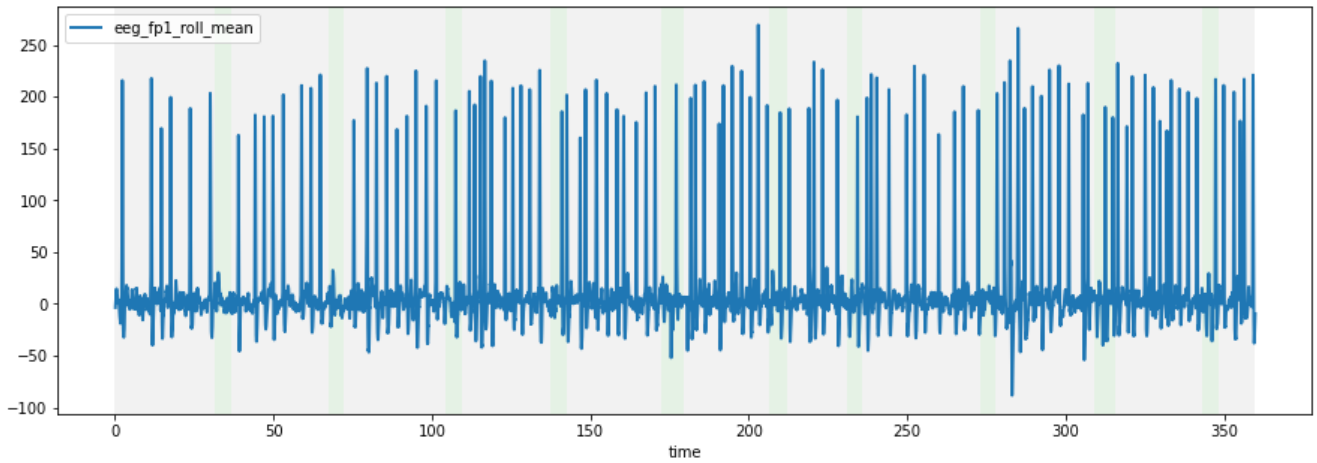

gsr_roll_mean
2000
1750
1500
1250
1000
750
500
250
0
0
50
100
150
200
250
300
350
time

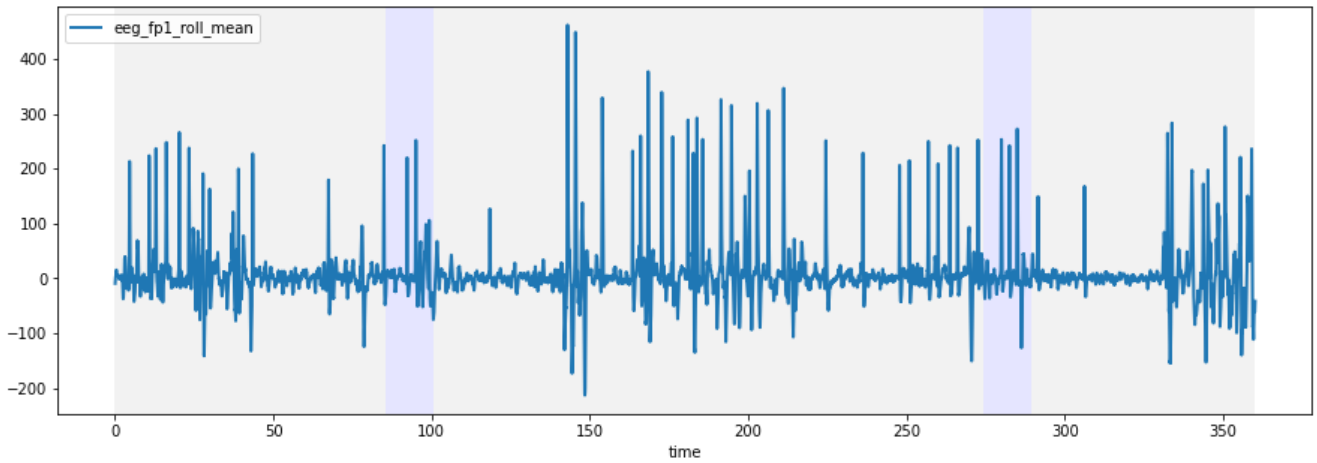

gsr_roll_mean
0.04
0.02
0.00
−0.02
−0.04
0
50
100
150
200
250
300
350
time

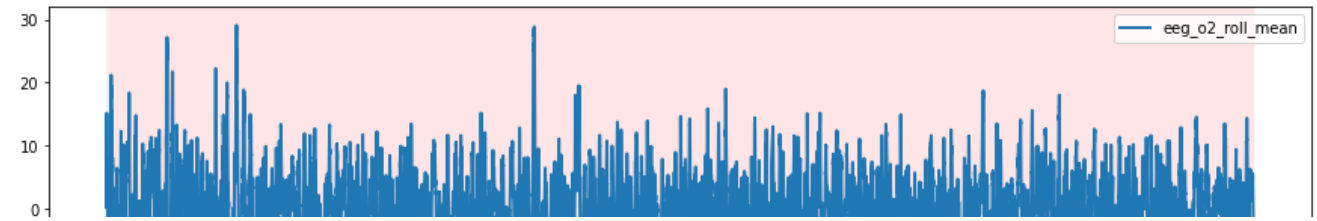

gsr_roll_mean
80
60

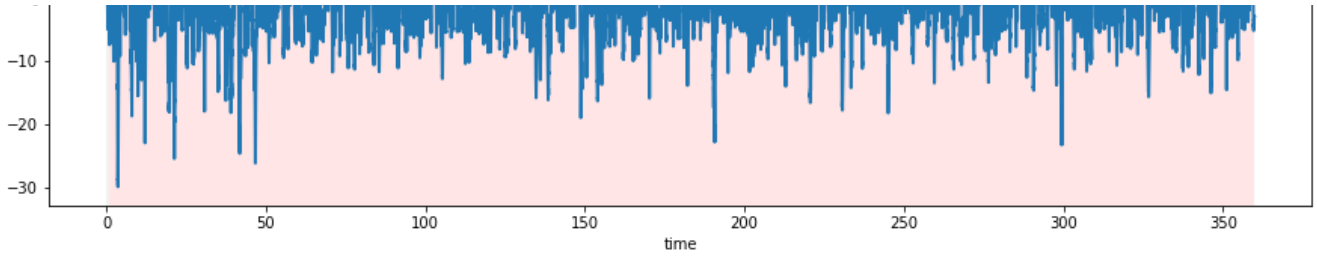

40
20
0
0
50
100
150
200
250
300
350
time

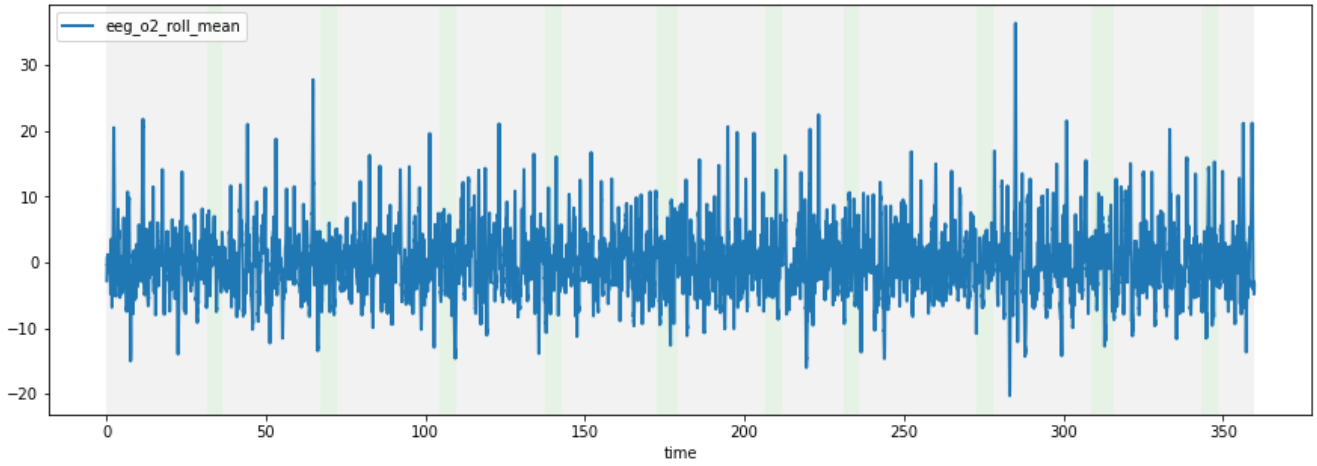

eeg_fp1_roll_mean
200
150
100
50
0
−50
−100
−150
0
50
100
150
200
250
300
350
time

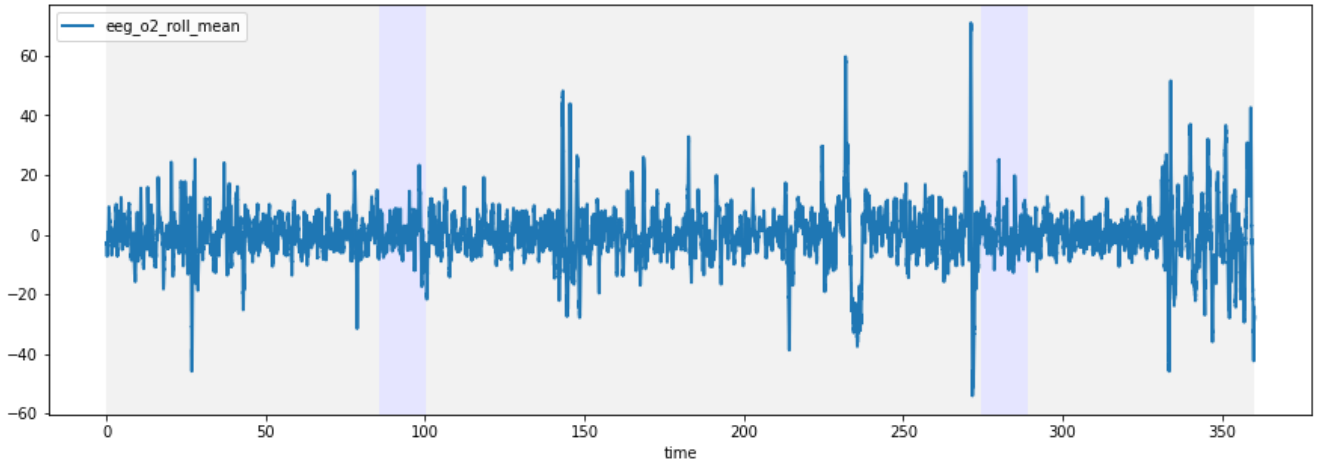

eeg_fp1_roll_mean
250
200
150
100
50
0
−50
−100
0
50
100
150
200
250
300
350
time


eeg_fp1_roll_mean
400
300
200
100
0
−100
−200
0
50
100
150
200
250
300
350
time


eeg_o2_roll_mean
30
20
10
0







In [44]:

```
calculate_values(df_tr_ti.loc[2,1],temp3)
```

```
Dataframe without Rolling Mean
+-----------------+---------------------+---------------------+---------------------+--------------
----+----------------------+---------------------+---------------------+---------------------+----
--------------+--------------------+---------------------+---------------------+---------------------
------------------+--------------------+---------------------+---------------------+--------------
--+---------------------+---------------------+---------------------+---------------------+----------
-------+
|      Value      |       eeg_fp1       |       eeg_f7        |       eeg_f8        |       eeg_t4
|         eeg_t6        |        eeg_t5       |        eeg_t3       |       eeg_fp2       |
g_o1       |       eeg_p3        |        eeg_pz       |       eeg_f3        |        eeg_fz       |
eeg_f4       |       eeg_c4        |        eeg_p4       |      eeg_poz        |       eeg_c3        |
eeg_cz       |        eeg_o2       |        ecg          |         r           |        gsr          |
+-----------------+---------------------+---------------------+---------------------+--------------
----+----------------------+---------------------+---------------------+---------------------+----
--------------+--------------------+---------------------+---------------------+---------------------
------------------+--------------------+---------------------+---------------------+--------------
--+---------------------+---------------------+---------------------+---------------------+----------
-------+
|       Mean      |  7.396034452744122  |  3.4358589017002825 |  1.4799443605730704 |
0.4717342029810818 |  -0.17040312181180023 | 0.10034965874244733 |  0.9499742576969004 |
6.658561990984308  | 0.17659826300061554 | 0.4991307207083674 | 0.48321869835751136 |
2.834035705343497  |  2.536396895318541  |  1.7972248985492385 | 0.7249466758764258 |
0.3144176387757382 | 0.2919161038204081 | 1.0754886545493905 |  0.9755117919720455 |
```

```
0.10401641055678021 | 8449.428587312937  | 814.1729804286042  | 206.99312814297303 |
|      Median      |  1.4115849999999999 | 0.43091199999999996 | 0.32016449999999996 |       0.16771
7      | -0.013379499999999999 |         0.0         |       0.04898       |       1.240425      |
-0.022516      |        0.0          |         0.0         | 0.5004919999999999 |  0.543806999999999
| 0.44695399999999996 |     0.1316345      |         0.0         |        0.0          |
0.1390985      | 0.11722949999999999 |         0.0         |    8537.320313     |     814.398987
|        0.0          |
|     Variance     |  2476.0039487279378 |  1203.1030982120312 |  495.5417530012894  |  246.61109120
231475 |   138.80282919085195  |  109.89136993563177 |  364.7548432523632  |  2086.807389397381  |
84.38162640722665  |  122.937211217108  |  144.20361112051884 | 634.7577578137032  |
499.81393208767156 |  418.35340536893534 | 196.85084187394966 |  128.8577706018921  |
112.01510826697906 | 231.9535723354181  |  213.78451249376528 |  128.0063529553846  |
1083602.5842152226 | 10.858856076005475 | 339553.9227830278  |
|       Max        |  579.2529910000002  |      415.096008     |  338.34899900000005 |      256.6530
5     |   246.94200099999998  |      153.145004     |      343.358002     |      561.708984     |
06.25700400000001 |     154.190994     |      109.072998     |     271.766998     |      246.878006
|  231.76400800000002 |     185.602005     |      138.951004     |     104.497002     |     194.264
99     |       139.638       |  129.08900500000001 |    11498.299805    |     828.35199      |    1
99.859985     |
|       Min        | -223.40800499999997 |     -267.402008     |       -233.166      | -
198.85699499999998 |  -384.10101299999997  |     -206.604004     | -240.94799799999998 | -
226.80499300000002 |     -142.891006     |    -134.404999     |     -151.886002     |    -
178.315994     | -174.76800500000004 |     -222.660995     |    -200.147995     |     -177.613007
|    -157.365005     |    -168.817993     |     -190.496994     |     -162.337997     |    3222.610
07     |     793.659973     |        0.0          |
| 25th percentile |  -8.831452500000001 |      -7.809275      |       -7.08469      |  -
5.668575000000001 |   -5.861364999999999  |      -4.622755      |       -5.27839      |      -
9.428705      |      -5.138965      |     -5.2003625     |  -5.822417500000001 | -6.836650000000001
|  -6.702040000000001 |  -6.680655000000001 | -5.960357499999999 | -5.8830800000000005 |     -5.72
48275     |     -5.448615      |      -6.194235      |  -5.879487500000001 |    7527.830078     |
812.148987     |        0.0          |
| 75th percentile |       11.3222       |       9.083125      |  7.985295000000001  |      6.235987
|       5.7096275       |      4.7048775      |      5.6637775      |  12.018049999999999 |       5
209275      | 5.443992499999999  |       6.15205       | 8.031995000000002  |  7.860290000000001  |
7.8552824999999995 |     6.4912325      |      6.1752775      |     5.9036375      |      6.048345
|      6.7258175      |      5.9721975      |    9322.700195     |     816.599976     |        0.0
|
|       IQR        |      20.1536525     |  16.892400000000002 |      15.069985      |      11.90456
5     |       11.5709925      |      9.3276325      |      10.9421675     |      21.446755      |
10.34824      |     10.644355      |  11.974467500000001 | 14.868645000000003 |  14.562330000000003
|      14.5359375     |      12.45159       |      12.0583575     |     11.628465      |
11.496960000000001 |      12.9200525     |      11.851685      | 1794.8701169999995 |
4.450988999999936  |        0.0          |
+-----------------+---------------------+---------------------+---------------------+--------------
----+----------------------+---------------------+---------------------+---------------------+----
-------------+--------------------+---------------------+---------------------+---------------------
--------------------+--------------------+---------------------+---------------------+--------------
--+----------------------+---------------------+---------------------+---------------------+----------
-------+
+-------+-------------+
| Value |    event    |
+-------+-------------+
|  Mode |    0   -1   |
|       | dtype: int8 |
+-------+-------------+


Dataframe with Rolling values
+-----------------+---------------------+---------------------+---------------------+--------------
----+----------------------+---------------------+---------------------+---------------------+------
------------+----------------------+---------------------+---------------------+-----------------
+---------------------+---------------------+---------------------+---------------------+---------
----------+---------------------+---------------------+---------------------+---------------------+-
---------------+
|      Value      |       eeg_fp1       |       eeg_f7        |       eeg_f8        |       eeg_t4
|       eeg_t6        |       eeg_t5        |       eeg_t3        |      eeg_fp2        |       ee
o1       |        eeg_p3       |       eeg_pz        |       eeg_f3        |       eeg_fz
eeg_f4       |       eeg_c4        |       eeg_p4        |       eeg_poz       |       eeg_c3
|       eeg_cz        |       eeg_o2        |       ecg         |         r         |       gsr
|
+-----------------+---------------------+---------------------+---------------------+--------------
----+----------------------+---------------------+---------------------+---------------------+------
------------+----------------------+---------------------+---------------------+-----------------
+---------------------+---------------------+---------------------+---------------------+---------
----------+---------------------+---------------------+---------------------+---------------------+-
---------------+
|       Mean      |  7.4018467792473075 |   3.43615821957541  |  1.4906963081270004 |
```

```
0.477553071053907  | -0.1659807712117614 | 0.09865176099784653  |  0.9505366357099906 |
6.671412961256106  | 0.17822418890556616 |   0.5005641063540806  |  0.4858085416038086  |
2.836055234093045  |  2.541372936028678  |  1.8052837740308194 |  0.7304202500746672 |
0.3179437931420704  |  0.2948718430903359  |  1.076825587231824  |  0.9787542399655772 | 0.10707863
681833193  | 8449.354148501157  | 814.1727616844015  | 207.1559484512277 |
|      Median      |  1.8081234499994507 |  0.6209204200000238 |  0.3687988900005559 |
0.28231159000000433 | -0.2290912399999785 | -0.12455752999989336 | 0.10412051000010442 |
1.5336259299990125 | -0.0764024099998129 | -0.017056670000064656 | -0.03726122999993689 |
0.7285324599994772 |  0.7604639900000936 |  0.6007041899998313 |  0.1101891099998903 |
0.020688439999828372 | -0.02390736999987868 |  0.2179345700000954 | 0.21677530000007755 | -
0.05845062000002281 | 8532.312646450024  | 814.4032499400039  |        0.0        |
|     Variance     |  1878.7714916672107 |  973.0227475639841  |  396.74541492520024 |  185.96689553
96118  |  57.275145578084526 |  70.09796556271976  |  307.15955486398906 | 1570.183654550709  |  4
2.387768790827835 |    80.8463879947543  |  98.53384288900519  |   485.68497413077  |  376.85931
49825694  |  319.3128923886642  |  140.5920384770996  |  73.78302265558312  |  66.30383589147812
|  166.27294855366446 |  155.92215578166633 |  47.206134840497825  | 1058168.4499394363 | 10.825119
431575317 | 339404.9518650284 |
|       Max        |  462.14435549999774 |  323.60040033999985 |  282.9972186600001  |
148.09767486000004 |  111.71190212000005 |  107.16112218000002  |  266.8204590200005  |
461.4905602999994  |     40.83192602     |   80.51243499999957  |  73.60110001999993  |  194.5434
4071999896 |  199.21109984000043 |  186.3956601199997  |  131.4202418399999  |   86.6421378599999
|  59.23499646000012  |  121.42094419999985 |  106.44823442000038 |  70.88557489999998  | 10660.3
76230280071 | 828.2034913599981  | 1999.213198179999 |
|       Min        | -213.08758050000205 | -241.35639915999997 | -150.43938135999986 | -177.47547973
999986 | -118.24541146000003 | -109.76980249999991  |  -221.1287594999996 | -170.9532198000002 |
-45.26852095999993 |   -76.27479088000047  |  -89.55693038000001  | -143.35898072000097 | -
119.26800031999949 | -127.75620006000041 | -101.32114128000043 |  -70.52317050000002  |  -
54.34728451999993  | -108.64751168000012 |  -105.2982461799995 |  -54.07565254000005  |
4130.751679660035  | 793.9315133999969  |        0.0        |
| 25th percentile |  -7.498262870001469 |  -6.391410970000118 |  -5.696226454999533 | -
3.8682460899999924 |  -3.333797374999989 | -2.9689289249999216  | -3.8506994249999185 | -
7.855101530000787 | -3.5049231549998403 |   -3.760594185000064  | -4.3076163799999865  |  -
5.576198449999965 |  -5.277998520000141 | -5.2197845950004504 |  -4.496868905000283 |  -
4.081613575000153  | -4.0307900900000035  |  -4.281491765000022  |  -4.816570189999773 |  -
3.447572820000019  | 7522.1141920350165 |  812.145513375009  |        0.0        |
| 75th percentile |  11.001783234999731 |  8.146523250000332  |   6.90090408000039  |  4.4936359799
99751  |  2.9661499550000463 |  2.978047255000033   |  4.352720605000134  | 11.593030184999385 |  3
.464437455000045  |   4.026710579999945   |   4.69530529500012   |  6.944249369999055  |  6.7260190
95000013  |   6.63354439999995  | 5.110355984999728  |  4.392369249999872   |  4.2662924750001405
|  4.860072849999945  |  5.391434975000358  |  3.498228774999978   | 9332.978837800032  | 816.59339
70350047  |        0.0        |
|       IQR        |   18.5000461050012  |  14.53793422000045  |  12.597130534999923 |
8.361882069999744  |  6.299947330000036  |  5.946976179999955  |  8.203420030000053  |
19.448131715000173 |  6.969360609999885  |   7.787304765000009   |  9.002921675000106   |  12.52044
781999902  |  12.004017615000155 |   11.8533289950004  |  9.607224890000012  |  8.473982825000025
|  8.297082565000144   |  9.141564614999968  |  10.20800516500013  |  6.945801594999997   | 1810.86
46457650157 | 4.447883659995682  |        0.0        |
+-----------------+--------------------+---------------------+---------------------+--------------
----+--------------------+---------------------+---------------------+---------------------+------
------------+----------------------+---------------------+---------------------+-----------------
+--------------------+--------------------+---------------------+---------------------+--------
----------+---------------------+---------------------+---------------------+---------------------+-
---------------+
+-------+-------------+
| Value |    event    |
+-------+-------------+
|  Mode |    0   -1   |
|       | dtype: int8 |
+-------+-------------+
```

Checking most frequent events in each window

Crew 3 seat 0

In [17]:

```
modeval = lambda x : stats.mode(x)[0]
```

In [34]:

```
temp_y = df_tr_ti.loc[3,0]['event']
temp_t = df_tr_ti.loc[3,0]['time']
temp2 = cal_roll(50,df_tr_ti.loc[3,0])
```

```
temp2['event'] = temp_y
temp2['time'] = temp_t
temp2
```

Out[34]:

	eeg_fp1_roll_mean	eeg_fp1_roll_median	eeg_fp1_roll_max	eeg_fp1_roll_min	eeg_fp1_roll_var	eeg_fp1_roll_
experiment						
0	-5.551450	-5.551450	-5.55145	-5.55145	NaN	-5.551450
0	-8.789925	-8.789925	-5.55145	-12.02840	20.975441	-10.409162
0	-10.151950	-12.028400	-5.55145	-12.87600	16.053057	-12.452200
0	-10.153562	-11.093400	-5.55145	-12.87600	10.702048	-12.240300
0	-9.743512	-10.158400	-5.55145	-12.87600	8.867243	-12.028400
...	...	...	...	...	...	...
2	2.750985	3.299650	13.81220	-14.04650	38.016278	-0.255008
2	2.703246	3.299650	13.81220	-14.04650	38.231487	-0.279969
2	2.611539	3.299650	13.81220	-14.04650	37.276422	-0.279969
2	2.520332	3.299650	12.99330	-14.04650	35.607501	-0.279969
2	2.511246	3.299650	12.99330	-14.04650	35.522792	-0.279969

276402 rows × 186 columns

In [41]:

```
temp_ev = temp2['event'].rolling(window=50).apply(modeval)
temp2['most_common_event'] = temp_ev
temp2['most_common_event'].fillna(-1,inplace=True)
temp2[['most_common_event']] = temp2[['most_common_event']].astype(int)
temp2.head()
```

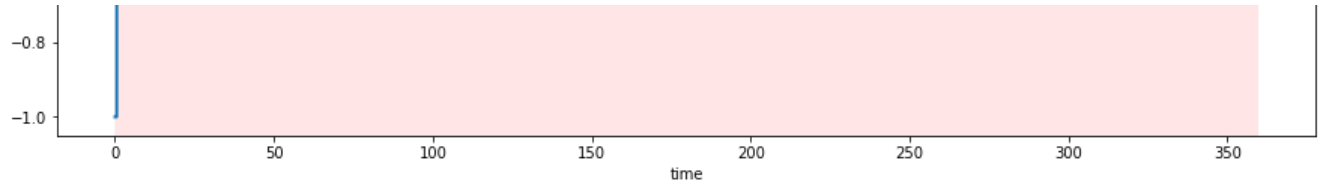
Out[41]:

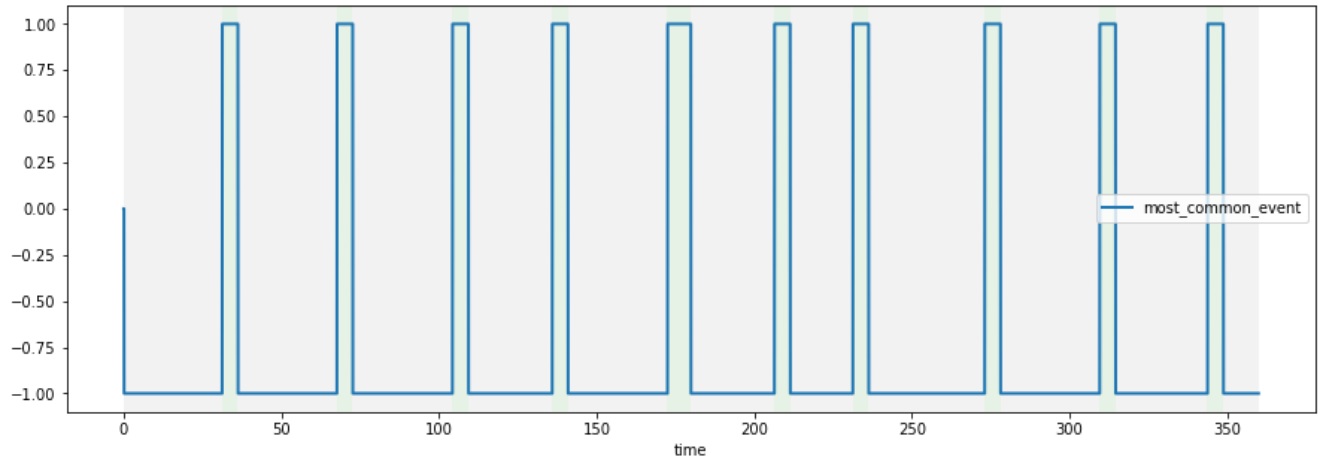
	eeg_fp1_roll_mean	eeg_fp1_roll_median	eeg_fp1_roll_max	eeg_fp1_roll_min	eeg_fp1_roll_var	eeg_fp1_roll_
experiment						
0	-5.551450	-5.551450	-5.55145	-5.55145	NaN	-5.551450
0	-8.789925	-8.789925	-5.55145	-12.02840	20.975441	-10.409162
0	-10.151950	-12.028400	-5.55145	-12.87600	16.053057	-12.452200
0	-10.153562	-11.093400	-5.55145	-12.87600	10.702048	-12.240300
0	-9.743512	-10.158400	-5.55145	-12.87600	8.867243	-12.028400

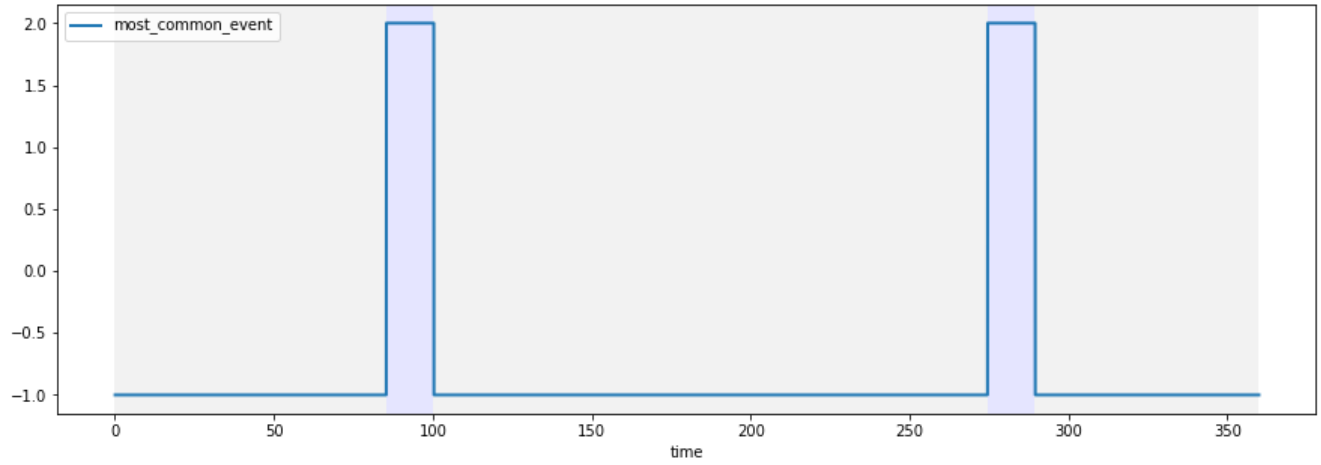
5 rows × 187 columns

In [43]:

```
get_plot(temp2,'most_common_event',0)
get_plot(temp2,'most_common_event',1)
get_plot(temp2,'most_common_event',2)
```







Crew 2 seat 1

In [44]:

```
temp_y = df_tr_ti.loc[2,1]['event']
temp_t = df_tr_ti.loc[2,1]['time']
temp2 = cal_roll(50,df_tr_ti.loc[2,1])
temp2['event'] = temp_y
temp2['time'] = temp_t
temp2
```

Out[44]:

	eeg_fp1_roll_mean	eeg_fp1_roll_median	eeg_fp1_roll_max	eeg_fp1_roll_min	eeg_fp1_roll_var	eeg_fp1_roll_
experiment						
0	28.280800	28.280800	28.280800	28.280800	NaN	28.280800
0	27.776649	27.776649	28.280800	27.272499	0.508335	27.524574
0	27.642799	27.375099	28.280800	27.272499	0.307915	27.323799
0	27.696149	27.615649	28.280800	27.272499	0.216662	27.349449
0	27.651120	27.471001	28.280800	27.272499	0.172635	27.375099
...	...	...	...	...	...	...
2	-44.296228	-45.300500	-21.860399	-74.690804	120.854914	-51.922825

2	-43.094890	-45.201851	-14.588900	-64.463900	118.525941	-50.085599
2	-42.068306	-44.619601	-13.099700	-62.003601	126.545201	-49.450623
2	-41.608144	-43.739099	-13.099700	-62.003601	121.707372	-49.182199
2	-41.456970	-43.739099	-13.099700	-62.003601	117.833216	-49.182199

276410 rows × 186 columns

In [45]:

```
temp_ev = temp2['event'].rolling(window=50).apply(modeval)
temp2['most_common_event'] = temp_ev
temp2['most_common_event'].fillna(-1,inplace=True)
temp2[['most_common_event']] = temp2[['most_common_event']].astype(int)
temp2.head()
```

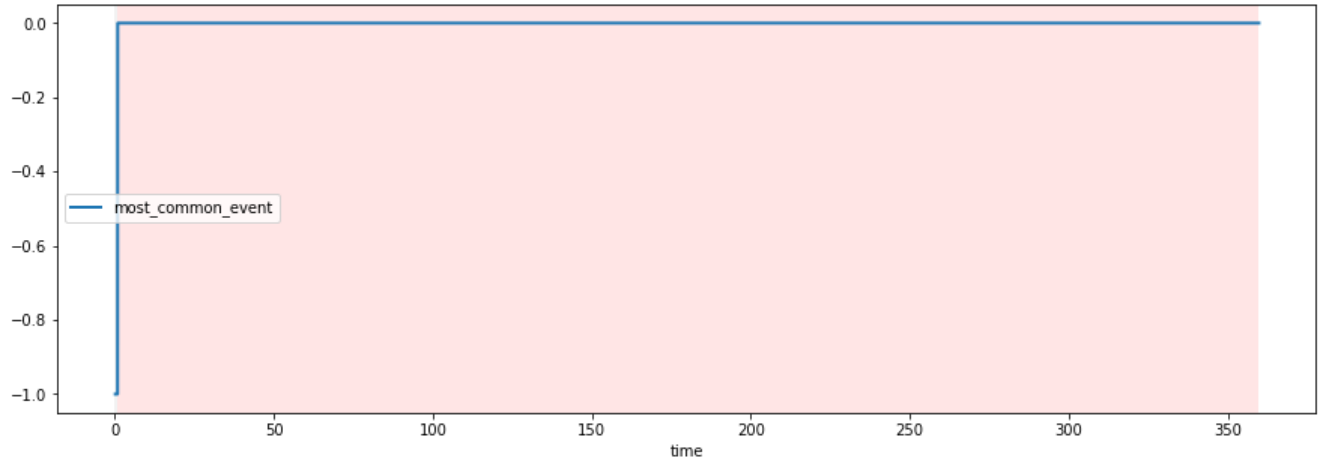
Out[45]:

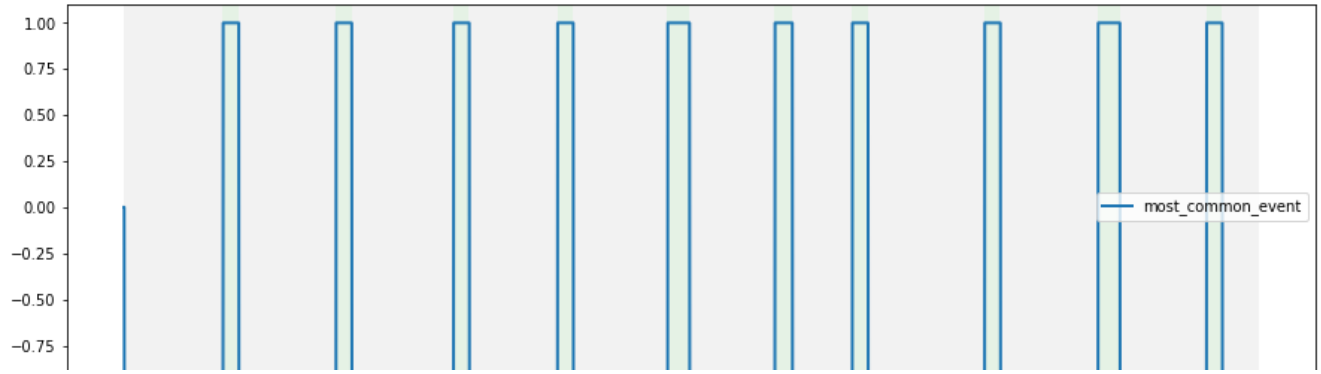
	eeg_fp1_roll_mean	eeg_fp1_roll_median	eeg_fp1_roll_max	eeg_fp1_roll_min	eeg_fp1_roll_var	eeg_fp1_roll_
experiment						
0	28.280800	28.280800	28.2808	28.280800	NaN	28.280800
0	27.776649	27.776649	28.2808	27.272499	0.508335	27.524574
0	27.642799	27.375099	28.2808	27.272499	0.307915	27.323799
0	27.696149	27.615649	28.2808	27.272499	0.216662	27.349449
0	27.651120	27.471001	28.2808	27.272499	0.172635	27.375099

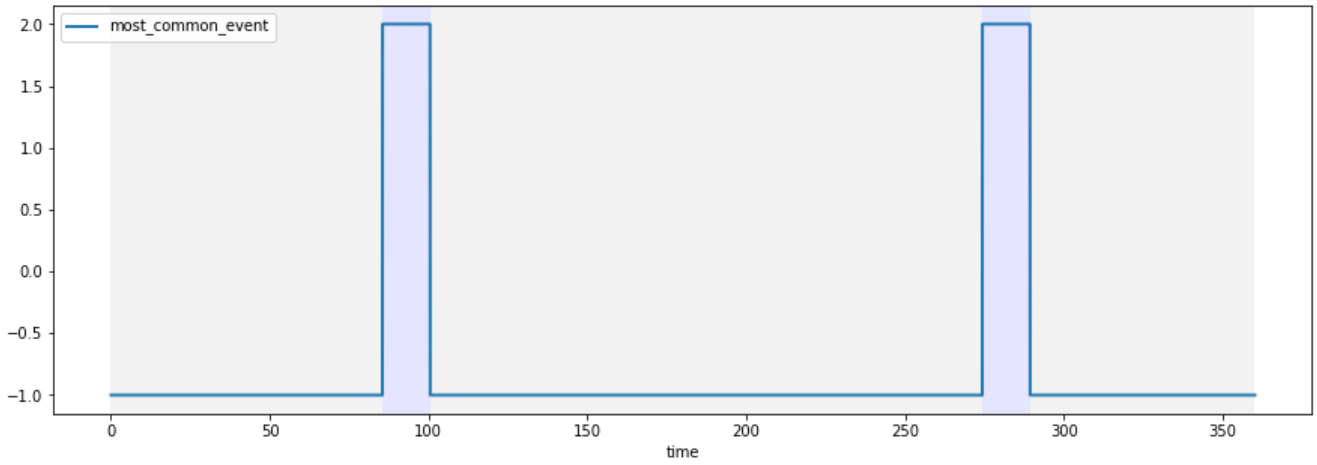
5 rows × 187 columns

In [46]:

```
get_plot(temp2,'most_common_event',0)
get_plot(temp2,'most_common_event',1)
get_plot(temp2,'most_common_event',2)
```







Feature Engineering

First let's add one feature pilot which will be a combination of crew and seat so that we can uniquely identify a pilot

In [7]:

```
df_train['pilot'] = 100 * df_train['seat'] + df_train['crew']
df_test['pilot'] = 100 * df_test['seat'] + df_test['crew']
```

In [13]:

```
df_train['experiment'] = df_train['experiment'].map({'CA': 0, 'DA': 1, 'SS': 2, 'LOFT': -1})
df_train["experiment"] = df_train["experiment"].astype('int8')
df_train['event'] = df_train['event'].map({'A': -1, 'B': 2, 'C': 0, 'D': 1})
df_train["event"] = df_train["event"].astype('int8')
y = df_train['event']

df_test['experiment'] = df_test['experiment'].map({'CA': 0, 'DA': 1, 'SS': 2, 'LOFT': -1})
df_test["experiment"] = df_test["experiment"].astype('int8')
```

In [8]:

```
df_train['experiment'] = df_train['experiment'].map({'CA': 0, 'DA': 1, 'SS': 2, 'LOFT': 3})
df_train["experiment"] = df_train["experiment"].astype('int8')
df_train['event'] = df_train['event'].map({'A': 3, 'B': 2, 'C': 0, 'D': 1})
df_train["event"] = df_train["event"].astype('int8')
y = df_train['event']

df_test['experiment'] = df_test['experiment'].map({'CA': 0, 'DA': 1, 'SS': 2, 'LOFT': 3})
df_test["experiment"] = df_test["experiment"].astype('int8')
```

In [53]:

```
df_train.head()
```

Out[53]:

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2
0	1	0	0.011719	1	-5.28545	26.775801	-9.527310	-12.793200	16.717800	33.737499	23.712299	-6.695870
1	1	0	0.015625	1	-2.42842	28.430901	-9.323510	-3.757230	15.969300	30.443600	21.010300	-6.474720
2	1	0	0.019531	1	10.67150	30.420200	15.350700	24.724001	16.143101	32.142799	25.431801	-0.088707

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	_eeg_fp2
3	1	0	0.023438	1	11.45250	25.609800	2.433080	12.412500	20.533300	31.494101	19.142799	0.256516
4	1	0	0.027344	1	7.28321	25.942600	0.113564	5.748000	19.833599	28.753599	20.572100	-1.953470

EEG Data

- The data is prepared in a fairly typical arrangement of 20 electrodes across the scalp. The letter in each lead signifies the part of the brain that that lead is nearest to (Temporal, Frontal, Parietal etc), with odd numbers on the left, evens on the right. Usually in the clinic, the electrical potentials at each electrode is not observed but the potential difference between pairs of electrodes.
- This gives us an idea of the electrical field in the brain region between these two points as a way to infer what the brain is doing in that region. We can choose any two electrodes and produce 20! different potential differences, but not all of those are going to be useful. *We talk about the layout of choosing the pairs of electrodes to compare potential differences as Montages. There's lots of different montage systems, but commonly there's the 10-20 system.

In [9]:

```
df_train['fp1_f7'] = df_train['eeg_fp1'] - df_train['eeg_f7']
df_train['f7_t3'] = df_train['eeg_f7'] - df_train['eeg_t3']
df_train['t3_t5'] = df_train['eeg_t3'] - df_train['eeg_t5']
df_train['t5_o1'] = df_train['eeg_t5'] - df_train['eeg_o1']
df_train['fp1_f3'] = df_train['eeg_fp1'] - df_train['eeg_f7']
df_train['f3_c3'] = df_train['eeg_f3'] - df_train['eeg_c3']
df_train['c3_p3'] = df_train['eeg_c3'] - df_train['eeg_p3']
df_train['p3_o1'] = df_train['eeg_p3'] - df_train['eeg_o1']
df_train['fz_cz'] = df_train['eeg_fz'] - df_train['eeg_cz']
df_train['cz_pz'] = df_train['eeg_cz'] - df_train['eeg_pz']
df_train['pz_poz'] = df_train['eeg_pz'] - df_train['eeg_poz']
df_train['fp2_f8'] = df_train['eeg_fp2'] - df_train['eeg_f8']
df_train['f8_t4'] = df_train['eeg_f8'] - df_train['eeg_t4']
df_train['t4_t6'] = df_train['eeg_t4'] - df_train['eeg_t6']
df_train['t6_o2'] = df_train['eeg_t6'] - df_train['eeg_o2']
df_train['fp2_f4'] = df_train['eeg_fp2'] - df_train['eeg_f4']
df_train['f4_c4'] = df_train['eeg_f4'] - df_train['eeg_c4']
df_train['c4_p4'] = df_train['eeg_c4'] - df_train['eeg_p4']
df_train['p4_o2'] = df_train['eeg_p4'] - df_train['eeg_o2']

new_features = ['fp1_f7','f7_t3','t3_t5','t5_o1','fp1_f3','f3_c3','c3_p3','p3_o1','fz_cz','cz_pz',
'pz_poz','fp2_f8','f8_t4','t4_t6','t6_o2','fp2_f4','f4_c4','c4_p4','p4_o2']
```

In [10]:

```
df_test['fp1_f7'] = df_test['eeg_fp1'] - df_test['eeg_f7']
df_test['f7_t3'] = df_test['eeg_f7'] - df_test['eeg_t3']
df_test['t3_t5'] = df_test['eeg_t3'] - df_test['eeg_t5']
df_test['t5_o1'] = df_test['eeg_t5'] - df_test['eeg_o1']
df_test['fp1_f3'] = df_test['eeg_fp1'] - df_test['eeg_f7']
df_test['f3_c3'] = df_test['eeg_f3'] - df_test['eeg_c3']
df_test['c3_p3'] = df_test['eeg_c3'] - df_test['eeg_p3']
df_test['p3_o1'] = df_test['eeg_p3'] - df_test['eeg_o1']
df_test['fz_cz'] = df_test['eeg_fz'] - df_test['eeg_cz']
df_test['cz_pz'] = df_test['eeg_cz'] - df_test['eeg_pz']
df_test['pz_poz'] = df_test['eeg_pz'] - df_test['eeg_poz']
df_test['fp2_f8'] = df_test['eeg_fp2'] - df_test['eeg_f8']
df_test['f8_t4'] = df_test['eeg_f8'] - df_test['eeg_t4']
df_test['t4_t6'] = df_test['eeg_t4'] - df_test['eeg_t6']
df_test['t6_o2'] = df_test['eeg_t6'] - df_test['eeg_o2']
df_test['fp2_f4'] = df_test['eeg_fp2'] - df_test['eeg_f4']
df_test['f4_c4'] = df_test['eeg_f4'] - df_test['eeg_c4']
df_test['c4_p4'] = df_test['eeg_c4'] - df_test['eeg_p4']
df_test['p4_o2'] = df_test['eeg_p4'] - df_test['eeg_o2']

new_features2 = ['fp1_f7','f7_t3','t3_t5','t5_o1','fp1_f3','f3_c3','c3_p3','p3_o1','fz_cz','cz_pz'
,'pz_poz','fp2_f8','f8_t4','t4_t6','t6_o2','fp2_f4','f4_c4','c4_p4','p4_o2']
```

In [14]:

```
df_train.head()
```

Out[14]:

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2
0	1	0	0.011719	1	-5.28545	26.775801	-9.527310	-12.793200	16.717800	33.737499	23.712299	-6.695870
1	1	0	0.015625	1	-2.42842	28.430901	-9.323510	-3.757230	15.969300	30.443600	21.010300	-6.474720
2	1	0	0.019531	1	10.67150	30.420200	15.350700	24.724001	16.143101	32.142799	25.431801	-0.088707
3	1	0	0.023438	1	11.45250	25.609800	2.433080	12.412500	20.533300	31.494101	19.142799	-0.256516
4	1	0	0.027344	1	7.28321	25.942600	0.113564	5.748000	19.833599	28.753599	20.572100	-1.953470

Since we have our features in various ranges and scale so we will normalize them

In [11]:

```
scaler = MinMaxScaler()
df_train[['eeg_fp1', 'eeg_f7', 'eeg_f8',
       'eeg_t4', 'eeg_t6', 'eeg_t5', 'eeg_t3', 'eeg_fp2', 'eeg_o1', 'eeg_p3',
       'eeg_pz', 'eeg_f3', 'eeg_fz', 'eeg_f4', 'eeg_c4', 'eeg_p4', 'eeg_poz',
       'eeg_c3', 'eeg_cz', 'eeg_o2', 'ecg', 'r', 'gsr',
       'fp1_f7', 'f7_t3', 't3_t5', 't5_o1', 'fp1_f3', 'f3_c3', 'c3_p3',
       'p3_o1', 'fz_cz', 'cz_pz', 'pz_poz', 'fp2_f8', 'f8_t4', 't4_t6',
       't6_o2', 'fp2_f4', 'f4_c4', 'c4_p4', 'p4_o2']] = scaler.fit_transform(df_train[['eeg_fp1', '
eeg_f7', 'eeg_f8',
       'eeg_t4', 'eeg_t6', 'eeg_t5', 'eeg_t3', 'eeg_fp2', 'eeg_o1', 'eeg_p3',
       'eeg_pz', 'eeg_f3', 'eeg_fz', 'eeg_f4', 'eeg_c4', 'eeg_p4', 'eeg_poz',
       'eeg_c3', 'eeg_cz', 'eeg_o2', 'ecg', 'r', 'gsr',
       'fp1_f7', 'f7_t3', 't3_t5', 't5_o1', 'fp1_f3', 'f3_c3', 'c3_p3',
       'p3_o1', 'fz_cz', 'cz_pz', 'pz_poz', 'fp2_f8', 'f8_t4', 't4_t6',
       't6_o2', 'fp2_f4', 'f4_c4', 'c4_p4', 'p4_o2']])
```

In [10]:

```
df_test.shape
```

Out[10]:

```
(17965143, 48)
```

In [12]:

```
dft1 = df_test.iloc[:5988381,:]
dft2 = df_test.iloc[5988381:11976762,:]
dft3 = df_test.iloc[11976762:17965143,:]
```

In [13]:

```
del df_test
import gc
gc.collect()
```

Out[13]:

```
0
```

In [14]:

```
scaler = MinMaxScaler()
dft1[['eeg_fp1', 'eeg_f7', 'eeg_f8',
      'eeg_t4', 'eeg_t6', 'eeg_t5', 'eeg_t3', 'eeg_fp2', 'eeg_o1', 'eeg_p3',
      'eeg_pz', 'eeg_f3', 'eeg_fz', 'eeg_f4', 'eeg_c4', 'eeg_p4', 'eeg_poz',
      'eeg_c3', 'eeg_cz', 'eeg_o2', 'ecg', 'r', 'gsr',
```

```
      'fp1_f7', 'f7_t3', 't3_t5', 't5_o1', 'fp1_f3', 'f3_c3', 'c3_p3',
      'p3_o1', 'fz_cz', 'cz_pz', 'pz_poz', 'fp2_f8', 'f8_t4', 't4_t6',
      't6_o2', 'fp2_f4', 'f4_c4', 'c4_p4', 'p4_o2']] = scaler.fit_transform(dft1[['eeg_fp1', 'eeg_f
7', 'eeg_f8',
      'eeg_t4', 'eeg_t6', 'eeg_t5', 'eeg_t3', 'eeg_fp2', 'eeg_o1', 'eeg_p3',
      'eeg_pz', 'eeg_f3', 'eeg_fz', 'eeg_f4', 'eeg_c4', 'eeg_p4', 'eeg_poz',
      'eeg_c3', 'eeg_cz', 'eeg_o2', 'ecg', 'r', 'gsr',
      'fp1_f7', 'f7_t3', 't3_t5', 't5_o1', 'fp1_f3', 'f3_c3', 'c3_p3',
      'p3_o1', 'fz_cz', 'cz_pz', 'pz_poz', 'fp2_f8', 'f8_t4', 't4_t6',
      't6_o2', 'fp2_f4', 'f4_c4', 'c4_p4', 'p4_o2']])

dft2[['eeg_fp1', 'eeg_f7', 'eeg_f8',
      'eeg_t4', 'eeg_t6', 'eeg_t5', 'eeg_t3', 'eeg_fp2', 'eeg_o1', 'eeg_p3',
      'eeg_pz', 'eeg_f3', 'eeg_fz', 'eeg_f4', 'eeg_c4', 'eeg_p4', 'eeg_poz',
      'eeg_c3', 'eeg_cz', 'eeg_o2', 'ecg', 'r', 'gsr',
      'fp1_f7', 'f7_t3', 't3_t5', 't5_o1', 'fp1_f3', 'f3_c3', 'c3_p3',
      'p3_o1', 'fz_cz', 'cz_pz', 'pz_poz', 'fp2_f8', 'f8_t4', 't4_t6',
      't6_o2', 'fp2_f4', 'f4_c4', 'c4_p4', 'p4_o2']] = scaler.fit_transform(dft2[['eeg_fp1', 'eeg_f
7', 'eeg_f8',
      'eeg_t4', 'eeg_t6', 'eeg_t5', 'eeg_t3', 'eeg_fp2', 'eeg_o1', 'eeg_p3',
      'eeg_pz', 'eeg_f3', 'eeg_fz', 'eeg_f4', 'eeg_c4', 'eeg_p4', 'eeg_poz',
      'eeg_c3', 'eeg_cz', 'eeg_o2', 'ecg', 'r', 'gsr',
      'fp1_f7', 'f7_t3', 't3_t5', 't5_o1', 'fp1_f3', 'f3_c3', 'c3_p3',
      'p3_o1', 'fz_cz', 'cz_pz', 'pz_poz', 'fp2_f8', 'f8_t4', 't4_t6',
      't6_o2', 'fp2_f4', 'f4_c4', 'c4_p4', 'p4_o2']])

dft3[['eeg_fp1', 'eeg_f7', 'eeg_f8',
      'eeg_t4', 'eeg_t6', 'eeg_t5', 'eeg_t3', 'eeg_fp2', 'eeg_o1', 'eeg_p3',
      'eeg_pz', 'eeg_f3', 'eeg_fz', 'eeg_f4', 'eeg_c4', 'eeg_p4', 'eeg_poz',
      'eeg_c3', 'eeg_cz', 'eeg_o2', 'ecg', 'r', 'gsr',
      'fp1_f7', 'f7_t3', 't3_t5', 't5_o1', 'fp1_f3', 'f3_c3', 'c3_p3',
      'p3_o1', 'fz_cz', 'cz_pz', 'pz_poz', 'fp2_f8', 'f8_t4', 't4_t6',
      't6_o2', 'fp2_f4', 'f4_c4', 'c4_p4', 'p4_o2']] = scaler.fit_transform(dft3[['eeg_fp1', 'eeg_f
7', 'eeg_f8',
      'eeg_t4', 'eeg_t6', 'eeg_t5', 'eeg_t3', 'eeg_fp2', 'eeg_o1', 'eeg_p3',
      'eeg_pz', 'eeg_f3', 'eeg_fz', 'eeg_f4', 'eeg_c4', 'eeg_p4', 'eeg_poz',
      'eeg_c3', 'eeg_cz', 'eeg_o2', 'ecg', 'r', 'gsr',
      'fp1_f7', 'f7_t3', 't3_t5', 't5_o1', 'fp1_f3', 'f3_c3', 'c3_p3',
      'p3_o1', 'fz_cz', 'cz_pz', 'pz_poz', 'fp2_f8', 'f8_t4', 't4_t6',
      't6_o2', 'fp2_f4', 'f4_c4', 'c4_p4', 'p4_o2']])
```

In [15]:

```
df_test_final = dft1.append([dft2, dft3], ignore_index = True)
```

In [16]:

```
del dft1 , dft2 , dft3
gc.collect()
```

Out[16]:

```
0
```

In [43]:

```
df_test_final.shape
```

Out[43]:

```
(17965143, 48)
```

In [44]:

```
df_train.head()
```

Out[44]:

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2	eeg_c
0	1	0	0.011719	1	0.406790	0.442990	0.431285	0.462933	0.583200	0.532208	0.466463	0.396589	0.6119

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2	eeg_o
1	1	0	0.015625	1	0.407647	0.443448	0.431339	0.465714	0.582847	0.530859	0.465496	0.396652	0.6143
2	1	0	0.019531	1	0.411577	0.443994	0.437849	0.474482	0.582929	0.531555	0.467078	0.398478	0.6116
3	1	0	0.023438	1	0.411811	0.442668	0.434441	0.470692	0.584998	0.531289	0.464828	0.398430	0.6122
4	1	0	0.027344	1	0.410560	0.442760	0.433829	0.468640	0.584669	0.530168	0.465339	0.397945	0.6124

In [17]:

```
df_test_id = df_test_final['id']
df_test_final = df_test_final.drop('id',axis=1)
```

In [27]:

```
df_test_final.head()
```

Out[27]:

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2	eeg_o
0	1	3	0.000000	0	0.495450	0.500571	0.493386	0.491813	0.489440	0.474928	0.491362	0.494778	0.5157
1	1	3	0.000000	1	0.498433	0.510128	0.495761	0.494974	0.494741	0.494422	0.495584	0.498799	0.4801
2	1	3	0.003906	0	0.497073	0.502968	0.492508	0.494004	0.491769	0.483483	0.496272	0.495359	0.5163
3	1	3	0.003906	1	0.498156	0.510251	0.495272	0.494673	0.494070	0.494114	0.495694	0.498150	0.4783
4	1	3	0.007812	0	0.494387	0.500284	0.492123	0.492009	0.491763	0.498938	0.489533	0.493982	0.5146

In [19]:

```
df_test_final.to_csv('test_data_processed.csv')
```

```
cp: cannot stat 'data.csv': No such file or directory
```

In [25]:

```
!cp "/content/test_data_processed.csv" "/content/gdrive/My Drive"
```

Let's train some algorithms

In [18]:

```
train, test = train_test_split(df_train, test_size=0.2, random_state=42, shuffle=True)
```

In [19]:

```
X_train = train.loc[:, df_train.columns != 'event']
y_train = train.event
# y_train = y_train.astype(int)

X_test = test.loc[:, df_train.columns != 'event']
y_test = test.event
# y_test = y_test.astype(int)

print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
(3893936, 47) (3893936,)
(973485, 47) (973485,)
```

Decision Tree model

In [79]:

```
param = {"max_depth" : [10, 50, 100, 200], "random_state" : [100]
```

```
param = {"max_depth" : [10,50,100,200], "random_state" : [100],
             "max_leaf_nodes" : [10,50,100,200],"criterion" : ['gini', 'entropy'],"max_features" :
['auto']}
model_dt = DecisionTreeClassifier()
random = RandomizedSearchCV(model_dt,param,verbose=50)
random.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=10, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=10, criterion=entropy,
score=0.722, total=  39.9s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   39.9s remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=10, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=10, criterion=entropy,
score=0.852, total=  39.2s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:  1.3min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=10, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=10, criterion=entropy,
score=0.835, total=  39.4s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:  2.0min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=10, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=10, criterion=entropy,
score=0.850, total=  39.2s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:  2.6min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=10, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=10, criterion=entropy,
score=0.856, total=  39.4s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:  3.3min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=entropy, s
core=0.819, total=  54.7s
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:  4.2min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=entropy, s
core=0.862, total=  44.1s
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:  4.9min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=entropy, s
core=0.870, total=  43.8s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:  5.7min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=entropy, s
core=0.882, total=  46.6s
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:  6.4min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=entropy, s
core=0.877, total=  52.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:  7.3min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=gini
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=gini,
score=0.803, total=  50.7s
[Parallel(n_jobs=1)]: Done  11 out of  11 | elapsed:  8.1min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=gini
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=gini,
score=0.860, total=  45.8s
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:  8.9min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=gini
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=gini,
score=0.852, total=  42.3s
[Parallel(n_jobs=1)]: Done  13 out of  13 | elapsed:  9.6min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=gini
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=gini,
score=0.816, total=  42.4s
[Parallel(n_jobs=1)]: Done  14 out of  14 | elapsed: 10.3min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=gini
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=200, criterion=gini,
score=0.839, total=  49.2s
[Parallel(n_jobs=1)]: Done  15 out of  15 | elapsed: 11.1min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=50, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=50, criterion=entropy,
score=0.819, total=  54.7s
[Parallel(n_jobs=1)]: Done  16 out of  16 | elapsed: 12.1min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=50, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=50, criterion=entropy,
score=0.862, total=  44.3s
[Parallel(n_jobs=1)]: Done  17 out of  17 | elapsed: 12.8min remaining:    0.0s
```

```
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=50, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=50, criterion=entropy,
score=0.870, total=  43.8s
[Parallel(n_jobs=1)]: Done  18 out of  18 | elapsed: 13.5min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=50, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=50, criterion=entropy,
score=0.882, total=  46.2s
[Parallel(n_jobs=1)]: Done  19 out of  19 | elapsed: 14.3min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=50, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=100, max_features=auto, max_depth=50, criterion=entropy,
score=0.877, total=  52.1s
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 15.2min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=50, criterion=gini
[CV]  random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=50, criterion=gini,
score=0.762, total=  44.7s
[Parallel(n_jobs=1)]: Done  21 out of  21 | elapsed: 15.9min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=50, criterion=gini
[CV]  random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=50, criterion=gini,
score=0.843, total=  43.5s
[Parallel(n_jobs=1)]: Done  22 out of  22 | elapsed: 16.6min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=50, criterion=gini
[CV]  random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=50, criterion=gini,
score=0.783, total=  38.1s
[Parallel(n_jobs=1)]: Done  23 out of  23 | elapsed: 17.3min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=50, criterion=gini
[CV]  random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=50, criterion=gini,
score=0.746, total=  37.4s
[Parallel(n_jobs=1)]: Done  24 out of  24 | elapsed: 17.9min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=50, criterion=gini
[CV]  random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=50, criterion=gini,
score=0.759, total=  46.4s
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed: 18.7min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=gini
[CV]  random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=gini,
score=0.683, total=  37.6s
[Parallel(n_jobs=1)]: Done  26 out of  26 | elapsed: 19.3min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=gini
[CV]  random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=gini,
score=0.686, total=  36.5s
[Parallel(n_jobs=1)]: Done  27 out of  27 | elapsed: 19.9min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=gini
[CV]  random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=gini,
score=0.738, total=  34.6s
[Parallel(n_jobs=1)]: Done  28 out of  28 | elapsed: 20.5min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=gini
[CV]  random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=gini,
score=0.796, total=  35.0s
[Parallel(n_jobs=1)]: Done  29 out of  29 | elapsed: 21.1min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=gini
[CV]  random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=gini,
score=0.683, total=  36.4s
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed: 21.7min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=100, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=100, criterion=entropy, s
core=0.852, total= 1.0min
[Parallel(n_jobs=1)]: Done  31 out of  31 | elapsed: 22.7min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=100, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=100, criterion=entropy, s
core=0.889, total=  53.2s
[Parallel(n_jobs=1)]: Done  32 out of  32 | elapsed: 23.6min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=100, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=100, criterion=entropy, s
core=0.888, total=  50.7s
[Parallel(n_jobs=1)]: Done  33 out of  33 | elapsed: 24.4min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=100, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=100, criterion=entropy, s
core=0.900, total=  51.3s
[Parallel(n_jobs=1)]: Done  34 out of  34 | elapsed: 25.3min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=100, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=100, criterion=entropy, s
core=0.890, total=  56.4s
[Parallel(n_jobs=1)]: Done  35 out of  35 | elapsed: 26.2min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=10, max_features=auto, max_depth=50, criterion=gini
[CV]  random_state=100, max_leaf_nodes=10, max_features=auto, max_depth=50, criterion=gini,
score=0.666, total=  18.7s
[Parallel(n_jobs=1)]: Done  36 out of  36 | elapsed: 26.5min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=10, max_features=auto, max_depth=50, criterion=gini
```

```
[CV]  random_state=100, max_leaf_nodes=10, max_features=auto, max_depth=50, criterion=gini,
score=0.666, total=  18.7s
[Parallel(n_jobs=1)]: Done  37 out of  37 | elapsed: 26.8min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=10, max_features=auto, max_depth=50, criterion=gini
[CV]  random_state=100, max_leaf_nodes=10, max_features=auto, max_depth=50, criterion=gini,
score=0.666, total=  17.1s
[Parallel(n_jobs=1)]: Done  38 out of  38 | elapsed: 27.1min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=10, max_features=auto, max_depth=50, criterion=gini
[CV]  random_state=100, max_leaf_nodes=10, max_features=auto, max_depth=50, criterion=gini,
score=0.665, total=  17.2s
[Parallel(n_jobs=1)]: Done  39 out of  39 | elapsed: 27.4min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=10, max_features=auto, max_depth=50, criterion=gini
[CV]  random_state=100, max_leaf_nodes=10, max_features=auto, max_depth=50, criterion=gini,
score=0.666, total=  18.7s
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed: 27.7min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=200, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=200, criterion=entropy, s
core=0.852, total= 1.0min
[Parallel(n_jobs=1)]: Done  41 out of  41 | elapsed: 28.7min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=200, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=200, criterion=entropy, s
core=0.889, total=  53.7s
[Parallel(n_jobs=1)]: Done  42 out of  42 | elapsed: 29.6min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=200, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=200, criterion=entropy, s
core=0.888, total=  50.9s
[Parallel(n_jobs=1)]: Done  43 out of  43 | elapsed: 30.5min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=200, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=200, criterion=entropy, s
core=0.900, total=  51.0s
[Parallel(n_jobs=1)]: Done  44 out of  44 | elapsed: 31.3min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=200, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=200, max_features=auto, max_depth=200, criterion=entropy, s
core=0.890, total=  55.3s
[Parallel(n_jobs=1)]: Done  45 out of  45 | elapsed: 32.3min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=entropy,
score=0.716, total=  37.7s
[Parallel(n_jobs=1)]: Done  46 out of  46 | elapsed: 32.9min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=entropy,
score=0.835, total=  36.8s
[Parallel(n_jobs=1)]: Done  47 out of  47 | elapsed: 33.5min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=entropy,
score=0.828, total=  37.4s
[Parallel(n_jobs=1)]: Done  48 out of  48 | elapsed: 34.1min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=entropy,
score=0.831, total=  37.0s
[Parallel(n_jobs=1)]: Done  49 out of  49 | elapsed: 34.7min remaining:    0.0s
[CV] random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=entropy
[CV]  random_state=100, max_leaf_nodes=50, max_features=auto, max_depth=10, criterion=entropy,
score=0.853, total=  37.2s
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed: 35.4min finished
```

Out[79]:

```
RandomizedSearchCV(cv=None, error_score=nan,
                   estimator=DecisionTreeClassifier(ccp_alpha=0.0,
                                                    class_weight=None,
                                                    criterion='gini',
                                                    max_depth=None,
                                                    max_features=None,
                                                    max_leaf_nodes=None,
                                                    min_impurity_decrease=0.0,
                                                    min_impurity_split=None,
                                                    min_samples_leaf=1,
                                                    min_samples_split=2,
                                                    min_weight_fraction_leaf=0.0,
                                                    presort='deprecated',
                                                    random_state=None,
                                                    splitter='best'),
                   iid='deprecated', n_iter=10, n_jobs=None,
                   param_distributions={'criterion': ['gini', 'entropy'],
                                        'max_depth': [10, 50, 100, 200],
                                        'max_features': ['auto'],
```

```
                                    'max_leaf_nodes': [10, 50, 100, 200],
                                    'random_state': [100]},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=False, scoring=None, verbose=50)
```

In [80]:

```
random.best_params_
```

Out[80]:

```
{'criterion': 'entropy',
 'max_depth': 100,
 'max_features': 'auto',
 'max_leaf_nodes': 200,
 'random_state': 100}
```

In [45]:

```
clf_dt = DecisionTreeClassifier(criterion='entropy',max_depth=100,max_leaf_nodes=200,random_state=
100)
clf_dt = clf_dt.fit(X_train,y_train)
```

In [46]:

```
joblib.dump(clf_dt, 'decision_tree.pkl')
```

Out[46]:

```
['decision_tree.pkl']
```

In [30]:

```
clf_dt = joblib.load('decision_tree.pkl')
```

In [47]:

```
predicted_dt = clf_dt.predict_proba(df_test_final)
```

In [48]:

```
predicted_dt
```

Out[48]:

```
array([[0.99351585, 0.00648415, 0.        , 0.        ],
       [0.99351585, 0.00648415, 0.        , 0.        ],
       [0.99351585, 0.00648415, 0.        , 0.        ],
       ...,
       [0.        , 1.        , 0.        , 0.        ],
       [0.        , 1.        , 0.        , 0.        ],
       [0.        , 1.        , 0.        , 0.        ]])
```

In [49]:

```
submission = pd.DataFrame(np.concatenate((np.arange(len(df_test_final))[:, np.newaxis],
predicted_dt), axis=1), columns=['id', 'A', 'B', 'C', 'D'])
submission['id'] = df_test_id.astype(int)
submission.to_csv("Submission.csv", index=False)
```

In [82]:

```
predicted_dt = clf_dt.predict_proba(X_test)
loss_dt = log_loss(y_test,predicted_dt)
print('Log loss = ',loss_dt)
```

```
Log loss =  0.037880678586169
```

Random Forest

In [84]:

```
param = {'n_estimators':[1,5,10,100],'max_depth' : [5,10,100], 'criterion' : ['gini','entropy'],'random_state' : [100],'n_jobs' : [-1]}
model_rf = RandomForestClassifier()
random_rf = RandomizedSearchCV(model_rf,param,verbose=10)
random_rf.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
```

```
[CV] random_state=100, n_jobs=-1, n_estimators=100, max_depth=5, criterion=entropy
```

```
exception calling callback for <Future at 0x7fc4e58b68d0 state=finished raised BrokenProcessPool>
joblib.externals.loky.process_executor._RemoteTraceback:
"""
Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py", line 404, in _process_worker
    call_item = call_queue.get(block=True, timeout=timeout)
  File "/usr/lib/python3.6/multiprocessing/queues.py", line 99, in get
    if not self._rlock.acquire(block, timeout):
KeyboardInterrupt
"""

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/_base.py", line 625, in _invoke_callbacks
    callback(self)
  File "/usr/local/lib/python3.6/dist-packages/joblib/parallel.py", line 347, in __call__
    self.parallel.dispatch_next()
  File "/usr/local/lib/python3.6/dist-packages/joblib/parallel.py", line 780, in dispatch_next
    if not self.dispatch_one_batch(self._original_iterator):
  File "/usr/local/lib/python3.6/dist-packages/joblib/parallel.py", line 847, in dispatch_one_batch
    self._dispatch(tasks)
  File "/usr/local/lib/python3.6/dist-packages/joblib/parallel.py", line 765, in _dispatch
    job = self._backend.apply_async(batch, callback=cb)
  File "/usr/local/lib/python3.6/dist-packages/joblib/_parallel_backends.py", line 531, in apply_async
    future = self._workers.submit(SafeFunction(func))
  File "/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/reusable_executor.py", line 178, in submit
    fn, *args, **kwargs)
  File "/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py", line 1102, in submit
    raise self._flags.broken
joblib.externals.loky.process_executor.BrokenProcessPool: A task has failed to un-serialize. Please ensure that the arguments of the function are all picklable.
```

```
[CV]  random_state=100, n_jobs=-1, n_estimators=100, max_depth=5, criterion=entropy, score=nan, total=   3.9s
[CV] random_state=100, n_jobs=-1, n_estimators=100, max_depth=5, criterion=entropy
```

```
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    3.9s remaining:    0.0s
```

```
[CV]  random_state=100, n_jobs=-1, n_estimators=100, max_depth=5, criterion=entropy, score=0.924, total= 1.1min
[CV] random_state=100, n_jobs=-1, n_estimators=100, max_depth=5, criterion=entropy
```

```
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:  1.2min remaining:    0.0s
```

```
[CV]  random_state=100, n_jobs=-1, n_estimators=100, max_depth=5, criterion=entropy, score=0.924, total= 1.0min
[CV] random_state=100, n_jobs=-1, n_estimators=100, max_depth=5, criterion=entropy
```

```
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:  2.2min remaining:    0.0s

[CV]  random_state=100, n_jobs=-1, n_estimators=100, max_depth=5, criterion=entropy, score=0.924,
total= 1.0min
[CV] random_state=100, n_jobs=-1, n_estimators=100, max_depth=5, criterion=entropy

[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:  3.2min remaining:    0.0s

[CV]  random_state=100, n_jobs=-1, n_estimators=100, max_depth=5, criterion=entropy, score=0.924,
total= 1.0min
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=entropy

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:  4.2min remaining:    0.0s

[CV]  random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=entropy, score=0.950,
total= 1.2min
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=entropy

[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:  5.4min remaining:    0.0s

[CV]  random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=entropy, score=0.958,
total= 1.2min
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=entropy

[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:  6.6min remaining:    0.0s

[CV]  random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=entropy, score=0.969,
total= 1.1min
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=entropy

[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:  7.7min remaining:    0.0s

[CV]  random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=entropy, score=0.952,
total= 1.3min
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=entropy

[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:  9.0min remaining:    0.0s

[CV]  random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=entropy, score=0.953,
total= 1.1min
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=5, criterion=gini
[CV]  random_state=100, n_jobs=-1, n_estimators=10, max_depth=5, criterion=gini, score=0.922, tota
l=  21.5s
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=5, criterion=gini
[CV]  random_state=100, n_jobs=-1, n_estimators=10, max_depth=5, criterion=gini, score=0.924, tota
l=  14.6s
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=5, criterion=gini
[CV]  random_state=100, n_jobs=-1, n_estimators=10, max_depth=5, criterion=gini, score=0.924, tota
l=  16.2s
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=5, criterion=gini
[CV]  random_state=100, n_jobs=-1, n_estimators=10, max_depth=5, criterion=gini, score=0.922, tota
l=  16.4s
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=5, criterion=gini
[CV]  random_state=100, n_jobs=-1, n_estimators=10, max_depth=5, criterion=gini, score=0.922, tota
l=  18.0s
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=100, criterion=entropy
[CV]  random_state=100, n_jobs=-1, n_estimators=10, max_depth=100, criterion=entropy, score=0.982,
total= 1.1min
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=100, criterion=entropy
[CV]  random_state=100, n_jobs=-1, n_estimators=10, max_depth=100, criterion=entropy, score=0.981,
total= 1.1min
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=100, criterion=entropy
[CV]  random_state=100, n_jobs=-1, n_estimators=10, max_depth=100, criterion=entropy, score=0.982,
total= 1.2min
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=100, criterion=entropy
[CV]  random_state=100, n_jobs=-1, n_estimators=10, max_depth=100, criterion=entropy, score=0.981,
total= 1.4min
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=100, criterion=entropy
[CV]  random_state=100, n_jobs=-1, n_estimators=10, max_depth=100, criterion=entropy, score=0.982,
```

[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=100, criterion=entropy, score=0.952,
total= 1.2min
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=10, criterion=entropy
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=10, criterion=entropy, score=0.926, t
otal= 33.5s
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=10, criterion=entropy
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=10, criterion=entropy, score=0.926, t
otal= 34.6s
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=10, criterion=entropy
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=10, criterion=entropy, score=0.926, t
otal= 35.5s
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=10, criterion=entropy
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=10, criterion=entropy, score=0.926, t
otal= 35.7s
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=10, criterion=entropy
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=10, criterion=entropy, score=0.926, t
otal= 34.0s
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=5, criterion=entropy
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=5, criterion=entropy, score=0.916, to
tal= 16.8s
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=5, criterion=entropy
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=5, criterion=entropy, score=0.916, to
tal= 16.6s
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=5, criterion=entropy
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=5, criterion=entropy, score=0.914, to
tal= 16.6s
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=5, criterion=entropy
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=5, criterion=entropy, score=0.913, to
tal= 16.7s
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=5, criterion=entropy
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=5, criterion=entropy, score=0.920, to
tal= 16.7s
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=10, criterion=entropy
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=10, criterion=entropy, score=0.926,
total= 33.5s
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=10, criterion=entropy
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=10, criterion=entropy, score=0.926,
total= 36.1s
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=10, criterion=entropy
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=10, criterion=entropy, score=0.925,
total= 36.2s
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=10, criterion=entropy
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=10, criterion=entropy, score=0.927,
total= 36.4s
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=10, criterion=entropy
[CV] random_state=100, n_jobs=-1, n_estimators=10, max_depth=10, criterion=entropy, score=0.925,
total= 34.4s
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=gini
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=gini, score=0.960, tot
al= 56.1s
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=gini
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=gini, score=0.950, tot
al= 56.9s
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=gini
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=gini, score=0.952, tot
al= 1.1min
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=gini
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=gini, score=0.956, tot
al= 1.0min
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=gini
[CV] random_state=100, n_jobs=-1, n_estimators=1, max_depth=100, criterion=gini, score=0.948, tot
al= 52.6s
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=5, criterion=gini
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=5, criterion=gini, score=0.923,
total= 18.9s
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=5, criterion=gini
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=5, criterion=gini, score=0.923,
total= 18.1s
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=5, criterion=gini
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=5, criterion=gini, score=0.924,
total= 13.2s
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=5, criterion=gini
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=5, criterion=gini, score=0.923,
total= 12.9s
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=5, criterion=gini
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=5, criterion=gini, score=0.923,
total= 13.2s
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=100, criterion=entropy

```
[CV]  random_state=100, n_jobs=-1, n_estimators=5, max_depth=100, criterion=entropy, score=0.982,
total= 1.2min
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=100, criterion=entropy
[CV]  random_state=100, n_jobs=-1, n_estimators=5, max_depth=100, criterion=entropy, score=0.982,
total=  58.3s
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=100, criterion=entropy
[CV]  random_state=100, n_jobs=-1, n_estimators=5, max_depth=100, criterion=entropy, score=0.983,
total= 1.2min
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=100, criterion=entropy
[CV]  random_state=100, n_jobs=-1, n_estimators=5, max_depth=100, criterion=entropy, score=0.982,
total= 1.4min
[CV] random_state=100, n_jobs=-1, n_estimators=5, max_depth=100, criterion=entropy
[CV]  random_state=100, n_jobs=-1, n_estimators=5, max_depth=100, criterion=entropy, score=0.979,
total= 1.1min

[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed: 36.8min finished
```

Out[84]:

```
RandomizedSearchCV(cv=None, error_score=nan,
                   estimator=RandomForestClassifier(bootstrap=True,
                                                    ccp_alpha=0.0,
                                                    class_weight=None,
                                                    criterion='gini',
                                                    max_depth=None,
                                                    max_features='auto',
                                                    max_leaf_nodes=None,
                                                    max_samples=None,
                                                    min_impurity_decrease=0.0,
                                                    min_impurity_split=None,
                                                    min_samples_leaf=1,
                                                    min_samples_split=2,
                                                    min_weight_fraction_leaf=0.0,
                                                    n_estimators=100,
                                                    n_jobs=None,
                                                    oob_score=False,
                                                    random_state=None,
                                                    verbose=0,
                                                    warm_start=False),
                   iid='deprecated', n_iter=10, n_jobs=None,
                   param_distributions={'criterion': ['gini', 'entropy'],
                                        'max_depth': [5, 10, 100],
                                        'n_estimators': [1, 5, 10, 100],
                                        'n_jobs': [-1], 'random_state': [100]},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=False, scoring=None, verbose=10)
```

In [86]:

```
random_rf.best_params_
```

Out[86]:

```
{'criterion': 'entropy',
 'max_depth': 100,
 'n_estimators': 10,
 'n_jobs': -1,
 'random_state': 100}
```

In [87]:

```
clf_rf = RandomForestClassifier(criterion='entropy',max_depth=100,n_estimators=10,random_state=100
)
clf_rf = clf_rf.fit(X_train,y_train)
```

In [88]:

```
predicted_rf = clf_rf.predict_proba(X_test)
loss_rf = log_loss(y_test,predicted_rf)
print('Log loss = ',loss_rf)
```

```
Log loss =  0.05680608782397054
```

Logistic Regression

In [89]:

```
param = {'penalty':['l1', 'l2', 'elasticnet', 'none'],'C': [0.01,0.1,1,10,100],'multi_class':['ovr'
,'multinomial'],'n_jobs' : [-1]}
model_lg = LogisticRegression()
random_lg = RandomizedSearchCV(model_lg,param,verbose=10)
random_lg.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

[CV] penalty=none, n_jobs=-1, multi_class=ovr, C=10 ..................
[CV]  penalty=none, n_jobs=-1, multi_class=ovr, C=10, score=0.924, total= 7.4min
[CV] penalty=none, n_jobs=-1, multi_class=ovr, C=10 ..................

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:  7.4min remaining:    0.0s

[CV]  penalty=none, n_jobs=-1, multi_class=ovr, C=10, score=0.924, total= 7.9min
[CV] penalty=none, n_jobs=-1, multi_class=ovr, C=10 ..................

[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed: 15.2min remaining:    0.0s

[CV]  penalty=none, n_jobs=-1, multi_class=ovr, C=10, score=0.924, total= 7.6min
[CV] penalty=none, n_jobs=-1, multi_class=ovr, C=10 ..................

[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed: 22.9min remaining:    0.0s

[CV]  penalty=none, n_jobs=-1, multi_class=ovr, C=10, score=0.924, total= 7.2min
[CV] penalty=none, n_jobs=-1, multi_class=ovr, C=10 ..................

[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed: 30.0min remaining:    0.0s

[CV]  penalty=none, n_jobs=-1, multi_class=ovr, C=10, score=0.924, total= 7.0min
[CV] penalty=l1, n_jobs=-1, multi_class=multinomial, C=0.1 ...........

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed: 37.1min remaining:    0.0s

[CV]  penalty=l1, n_jobs=-1, multi_class=multinomial, C=0.1, score=nan, total=   0.7s
[CV] penalty=l1, n_jobs=-1, multi_class=multinomial, C=0.1 ...........

[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed: 37.1min remaining:    0.0s

[CV]  penalty=l1, n_jobs=-1, multi_class=multinomial, C=0.1, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=multinomial, C=0.1 ...........

[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed: 37.1min remaining:    0.0s

[CV]  penalty=l1, n_jobs=-1, multi_class=multinomial, C=0.1, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=multinomial, C=0.1 ...........

[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed: 37.1min remaining:    0.0s

[CV]  penalty=l1, n_jobs=-1, multi_class=multinomial, C=0.1, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=multinomial, C=0.1 ...........

[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed: 37.1min remaining:    0.0s

[CV]  penalty=l1, n_jobs=-1, multi_class=multinomial, C=0.1, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=ovr, C=0.01 ..................
[CV]  penalty=l1, n_jobs=-1, multi_class=ovr, C=0.01, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=ovr, C=0.01
```

```
[CV] penalty=l1, n_jobs=-1, multi_class=ovr, C=0.01 ..................
[CV]  penalty=l1, n_jobs=-1, multi_class=ovr, C=0.01, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=ovr, C=0.01 ..................
[CV]  penalty=l1, n_jobs=-1, multi_class=ovr, C=0.01, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=ovr, C=0.01 ..................
[CV]  penalty=l1, n_jobs=-1, multi_class=ovr, C=0.01, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=ovr, C=0.01 ..................
[CV]  penalty=l1, n_jobs=-1, multi_class=ovr, C=0.01, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=ovr, C=0.1 ...................
[CV]  penalty=l1, n_jobs=-1, multi_class=ovr, C=0.1, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=ovr, C=0.1 ...................
[CV]  penalty=l1, n_jobs=-1, multi_class=ovr, C=0.1, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=ovr, C=0.1 ...................
[CV]  penalty=l1, n_jobs=-1, multi_class=ovr, C=0.1, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=ovr, C=0.1 ...................
[CV]  penalty=l1, n_jobs=-1, multi_class=ovr, C=0.1, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=ovr, C=0.1 ...................
[CV]  penalty=l1, n_jobs=-1, multi_class=ovr, C=0.1, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=ovr, C=10 ....................
[CV]  penalty=l1, n_jobs=-1, multi_class=ovr, C=10, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=ovr, C=10 ....................
[CV]  penalty=l1, n_jobs=-1, multi_class=ovr, C=10, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=ovr, C=10 ....................
[CV]  penalty=l1, n_jobs=-1, multi_class=ovr, C=10, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=ovr, C=10 ....................
[CV]  penalty=l1, n_jobs=-1, multi_class=ovr, C=10, score=nan, total=   0.5s
[CV] penalty=l1, n_jobs=-1, multi_class=ovr, C=10 ....................
[CV]  penalty=l1, n_jobs=-1, multi_class=ovr, C=10, score=nan, total=   0.5s
[CV] penalty=none, n_jobs=-1, multi_class=multinomial, C=100 .........
[CV]  penalty=none, n_jobs=-1, multi_class=multinomial, C=100, score=0.924, total= 3.9min
[CV] penalty=none, n_jobs=-1, multi_class=multinomial, C=100 .........
[CV]  penalty=none, n_jobs=-1, multi_class=multinomial, C=100, score=0.924, total= 3.9min
[CV] penalty=none, n_jobs=-1, multi_class=multinomial, C=100 .........
[CV]  penalty=none, n_jobs=-1, multi_class=multinomial, C=100, score=0.924, total= 3.9min
[CV] penalty=none, n_jobs=-1, multi_class=multinomial, C=100 .........
[CV]  penalty=none, n_jobs=-1, multi_class=multinomial, C=100, score=0.924, total= 3.7min
[CV] penalty=none, n_jobs=-1, multi_class=multinomial, C=100 .........
[CV]  penalty=none, n_jobs=-1, multi_class=multinomial, C=100, score=0.924, total= 4.0min
[CV] penalty=none, n_jobs=-1, multi_class=ovr, C=1 ...................
[CV]  penalty=none, n_jobs=-1, multi_class=ovr, C=1, score=0.924, total= 7.2min
[CV] penalty=none, n_jobs=-1, multi_class=ovr, C=1 ...................
[CV]  penalty=none, n_jobs=-1, multi_class=ovr, C=1, score=0.924, total= 6.9min
[CV] penalty=none, n_jobs=-1, multi_class=ovr, C=1 ...................
[CV]  penalty=none, n_jobs=-1, multi_class=ovr, C=1, score=0.924, total= 7.4min
[CV] penalty=none, n_jobs=-1, multi_class=ovr, C=1 ...................
[CV]  penalty=none, n_jobs=-1, multi_class=ovr, C=1, score=0.924, total= 7.1min
[CV] penalty=none, n_jobs=-1, multi_class=ovr, C=1 ...................
[CV]  penalty=none, n_jobs=-1, multi_class=ovr, C=1, score=0.924, total= 7.1min
[CV] penalty=l2, n_jobs=-1, multi_class=multinomial, C=1 ............
[CV]  penalty=l2, n_jobs=-1, multi_class=multinomial, C=1, score=0.924, total= 3.9min
[CV] penalty=l2, n_jobs=-1, multi_class=multinomial, C=1 ............
[CV]  penalty=l2, n_jobs=-1, multi_class=multinomial, C=1, score=0.924, total= 3.8min
[CV] penalty=l2, n_jobs=-1, multi_class=multinomial, C=1 ............
[CV]  penalty=l2, n_jobs=-1, multi_class=multinomial, C=1, score=0.924, total= 3.9min
[CV] penalty=l2, n_jobs=-1, multi_class=multinomial, C=1 ............
[CV]  penalty=l2, n_jobs=-1, multi_class=multinomial, C=1, score=0.924, total= 3.8min
[CV] penalty=l2, n_jobs=-1, multi_class=multinomial, C=1 ............
[CV]  penalty=l2, n_jobs=-1, multi_class=multinomial, C=1, score=0.924, total= 4.1min
[CV] penalty=none, n_jobs=-1, multi_class=multinomial, C=10 ..........
[CV]  penalty=none, n_jobs=-1, multi_class=multinomial, C=10, score=0.924, total= 3.8min
[CV] penalty=none, n_jobs=-1, multi_class=multinomial, C=10 ..........
[CV]  penalty=none, n_jobs=-1, multi_class=multinomial, C=10, score=0.924, total= 4.0min
[CV] penalty=none, n_jobs=-1, multi_class=multinomial, C=10 ..........
[CV]  penalty=none, n_jobs=-1, multi_class=multinomial, C=10, score=0.924, total= 3.7min
[CV] penalty=none, n_jobs=-1, multi_class=multinomial, C=10 ..........
[CV]  penalty=none, n_jobs=-1, multi_class=multinomial, C=10, score=0.924, total= 3.8min
[CV] penalty=none, n_jobs=-1, multi_class=multinomial, C=10 ..........
[CV]  penalty=none, n_jobs=-1, multi_class=multinomial, C=10, score=0.924, total= 3.9min
[CV] penalty=l2, n_jobs=-1, multi_class=ovr, C=0.1 ...................
[CV]  penalty=l2, n_jobs=-1, multi_class=ovr, C=0.1, score=0.924, total= 7.3min
[CV] penalty=l2, n_jobs=-1, multi_class=ovr, C=0.1 ...................
[CV]  penalty=l2, n_jobs=-1, multi_class=ovr, C=0.1, score=0.924, total= 7.8min
[CV] penalty=l2, n_jobs=-1, multi_class=ovr, C=0.1 ...................
[CV]  penalty=l2, n_jobs=-1, multi_class=ovr, C=0.1, score=0.924, total= 7.0min
[CV] penalty=l2, n_jobs=-1, multi_class=ovr, C=0.1 ...................
[CV]  penalty=l2, n_jobs=-1, multi_class=ovr, C=0.1, score=0.924, total= 7.4min
[CV] penalty=l2, n_jobs=-1, multi_class=ovr, C=0.1 ...................
```

```
[CV]  penalty=l2, n_jobs=-1, multi_class=ovr, C=0.1, score=0.924, total=  7.4min

[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed: 167.6min finished
```

Out[89]:

```
RandomizedSearchCV(cv=None, error_score=nan,
                   estimator=LogisticRegression(C=1.0, class_weight=None,
                                                dual=False, fit_intercept=True,
                                                intercept_scaling=1,
                                                l1_ratio=None, max_iter=100,
                                                multi_class='auto', n_jobs=None,
                                                penalty='l2', random_state=None,
                                                solver='lbfgs', tol=0.0001,
                                                verbose=0, warm_start=False),
                   iid='deprecated', n_iter=10, n_jobs=None,
                   param_distributions={'C': [0.01, 0.1, 1, 10, 100],
                                        'multi_class': ['ovr', 'multinomial'],
                                        'n_jobs': [-1],
                                        'penalty': ['l1', 'l2', 'elasticnet',
                                                    'none']},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=False, scoring=None, verbose=10)
```

In [91]:

```
random_lg.best_params_
```

Out[91]:

```
{'C': 10, 'multi_class': 'ovr', 'n_jobs': -1, 'penalty': 'none'}
```

In [92]:

```
clf_lg = LogisticRegression(C=10,multi_class='ovr',penalty='l2',random_state=100)
clf_lg = clf_lg.fit(X_train,y_train)
```

In [93]:

```
predicted_lg = clf_lg.predict_proba(X_test)
loss_lg = log_loss(y_test,predicted_lg)
print('Log loss = ',loss_lg)
```

```
Log loss =  0.29901234963328643
```

LightGBM

In [7]:

```
!git clone --recursive https://github.com/Microsoft/LightGBM
```

```
Cloning into 'LightGBM'...
remote: Enumerating objects: 47, done.
remote: Counting objects: 100% (47/47), done.
remote: Compressing objects: 100% (46/46), done.
remote: Total 19367 (delta 13), reused 0 (delta 0), pack-reused 19320
Receiving objects: 100% (19367/19367), 15.65 MiB | 14.29 MiB/s, done.
Resolving deltas: 100% (14124/14124), done.
Submodule 'include/boost/compute' (https://github.com/boostorg/compute) registered for path
'compute'
Cloning into '/content/LightGBM/compute'...
remote: Enumerating objects: 21728, done.
remote: Total 21728 (delta 0), reused 0 (delta 0), pack-reused 21728
Receiving objects: 100% (21728/21728), 8.51 MiB | 8.48 MiB/s, done.
Resolving deltas: 100% (17565/17565), done.
Submodule path 'compute': checked out '36c89134d4013b2e5e45bc55656a18bd6141995a'
```

In [15]:

```
%cd LightGBM
```

```
!mkdir build
%cd build
```

```
/content/LightGBM
/content/LightGBM/build
```

In [17]:

```
!cmake -DUSE_GPU=1 ..
!make -j4
```

```
-- The C compiler identification is GNU 7.5.0
-- The CXX compiler identification is GNU 7.5.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found OpenMP_C: -fopenmp (found version "4.5")
-- Found OpenMP_CXX: -fopenmp (found version "4.5")
-- Found OpenMP: TRUE (found version "4.5")
-- Looking for CL_VERSION_2_2
-- Looking for CL_VERSION_2_2 - found
-- Found OpenCL: /usr/lib/x86_64-linux-gnu/libOpenCL.so (found version "2.2")
-- OpenCL include directory: /usr/include
-- Boost version: 1.65.1
-- Found the following Boost libraries:
--   filesystem
--   system
-- Performing Test MM_PREFETCH
-- Performing Test MM_PREFETCH - Success
-- Using _mm_prefetch
-- Performing Test MM_MALLOC
-- Performing Test MM_MALLOC - Success
-- Using _mm_malloc
-- Configuring done
-- Generating done
-- Build files have been written to: /content/LightGBM/build
Scanning dependencies of target _lightgbm
Scanning dependencies of target lightgbm
[  1%] Building CXX object CMakeFiles/lightgbm.dir/src/main.cpp.o
[  3%] Building CXX object CMakeFiles/lightgbm.dir/src/application/application.cpp.o
[  4%] Building CXX object CMakeFiles/_lightgbm.dir/src/boosting/boosting.cpp.o
[  6%] Building CXX object CMakeFiles/_lightgbm.dir/src/application/application.cpp.o
[  7%] Building CXX object CMakeFiles/_lightgbm.dir/src/boosting/gbdt.cpp.o
[  9%] Building CXX object CMakeFiles/_lightgbm.dir/src/boosting/gbdt_model_text.cpp.o
[ 10%] Building CXX object CMakeFiles/lightgbm.dir/src/boosting/boosting.cpp.o
[ 12%] Building CXX object CMakeFiles/_lightgbm.dir/src/boosting/gbdt_prediction.cpp.o
[ 14%] Building CXX object CMakeFiles/_lightgbm.dir/src/boosting/prediction_early_stop.cpp.o
[ 15%] Building CXX object CMakeFiles/_lightgbm.dir/src/io/bin.cpp.o
[ 17%] Building CXX object CMakeFiles/_lightgbm.dir/src/io/config.cpp.o
[ 18%] Building CXX object CMakeFiles/_lightgbm.dir/src/io/config_auto.cpp.o
[ 20%] Building CXX object CMakeFiles/_lightgbm.dir/src/io/dataset.cpp.o
[ 21%] Building CXX object CMakeFiles/_lightgbm.dir/src/io/dataset_loader.cpp.o
[ 23%] Building CXX object CMakeFiles/lightgbm.dir/src/boosting/gbdt.cpp.o
[ 25%] Building CXX object CMakeFiles/lightgbm.dir/src/boosting/gbdt_model_text.cpp.o
[ 26%] Building CXX object CMakeFiles/_lightgbm.dir/src/io/file_io.cpp.o
[ 28%] Building CXX object CMakeFiles/_lightgbm.dir/src/io/json11.cpp.o
[ 29%] Building CXX object CMakeFiles/_lightgbm.dir/src/io/metadata.cpp.o
[ 31%] Building CXX object CMakeFiles/_lightgbm.dir/src/io/parser.cpp.o
[ 32%] Building CXX object CMakeFiles/_lightgbm.dir/src/io/tree.cpp.o
[ 34%] Building CXX object CMakeFiles/lightgbm.dir/src/boosting/gbdt_prediction.cpp.o
[ 35%] Building CXX object CMakeFiles/_lightgbm.dir/src/metric/dcg_calculator.cpp.o
[ 37%] Building CXX object CMakeFiles/lightgbm.dir/src/boosting/prediction_early_stop.cpp.o
[ 39%] Building CXX object CMakeFiles/_lightgbm.dir/src/metric/metric.cpp.o
[ 40%] Building CXX object CMakeFiles/lightgbm.dir/src/io/bin.cpp.o
[ 42%] Building CXX object CMakeFiles/_lightgbm.dir/src/network/linker_topo.cpp.o
[ 43%] Building CXX object CMakeFiles/_lightgbm.dir/src/network/linkers_mpi.cpp.o
```

```
[ 45%] Building CXX object CMakeFiles/_lightgbm.dir/src/network/linkers_socket.cpp.o
[ 46%] Building CXX object CMakeFiles/_lightgbm.dir/src/network/network.cpp.o
[ 48%] Building CXX object CMakeFiles/_lightgbm.dir/src/objective/objective_function.cpp.o
[ 50%] Building CXX object CMakeFiles/_lightgbm.dir/src/treelearner/cuda_tree_learner.cpp.o
[ 51%] Building CXX object
CMakeFiles/_lightgbm.dir/src/treelearner/data_parallel_tree_learner.cpp.o
[ 53%] Building CXX object
CMakeFiles/_lightgbm.dir/src/treelearner/feature_parallel_tree_learner.cpp.o
[ 54%] Building CXX object CMakeFiles/_lightgbm.dir/src/treelearner/gpu_tree_learner.cpp.o
[ 56%] Building CXX object CMakeFiles/_lightgbm.dir/src/treelearner/serial_tree_learner.cpp.o
[ 57%] Building CXX object CMakeFiles/_lightgbm.dir/src/treelearner/tree_learner.cpp.o
[ 59%] Building CXX object CMakeFiles/lightgbm.dir/src/io/config.cpp.o
[ 60%] Building CXX object CMakeFiles/lightgbm.dir/src/io/config_auto.cpp.o
[ 62%] Building CXX object
CMakeFiles/_lightgbm.dir/src/treelearner/voting_parallel_tree_learner.cpp.o
[ 64%] Building CXX object CMakeFiles/lightgbm.dir/src/io/dataset.cpp.o
[ 65%] Building CXX object CMakeFiles/_lightgbm.dir/src/c_api.cpp.o
[ 67%] Building CXX object CMakeFiles/lightgbm.dir/src/io/dataset_loader.cpp.o
[ 68%] Building CXX object CMakeFiles/lightgbm.dir/src/io/file_io.cpp.o
[ 70%] Building CXX object CMakeFiles/lightgbm.dir/src/io/json11.cpp.o
[ 71%] Building CXX object CMakeFiles/lightgbm.dir/src/io/metadata.cpp.o
[ 73%] Building CXX object CMakeFiles/lightgbm.dir/src/io/parser.cpp.o
[ 75%] Building CXX object CMakeFiles/lightgbm.dir/src/io/tree.cpp.o
[ 76%] Building CXX object CMakeFiles/lightgbm.dir/src/metric/dcg_calculator.cpp.o
[ 78%] Building CXX object CMakeFiles/lightgbm.dir/src/metric/metric.cpp.o
[ 79%] Building CXX object CMakeFiles/lightgbm.dir/src/network/linker_topo.cpp.o
[ 81%] Building CXX object CMakeFiles/lightgbm.dir/src/network/linkers_mpi.cpp.o
[ 82%] Building CXX object CMakeFiles/lightgbm.dir/src/network/linkers_socket.cpp.o
[ 84%] Building CXX object CMakeFiles/lightgbm.dir/src/network/network.cpp.o
[ 85%] Building CXX object CMakeFiles/lightgbm.dir/src/objective/objective_function.cpp.o
[ 87%] Building CXX object CMakeFiles/lightgbm.dir/src/treelearner/cuda_tree_learner.cpp.o
[ 89%] Building CXX object
CMakeFiles/lightgbm.dir/src/treelearner/data_parallel_tree_learner.cpp.o
[ 90%] Building CXX object
CMakeFiles/lightgbm.dir/src/treelearner/feature_parallel_tree_learner.cpp.o
[ 92%] Linking CXX shared library ../lib_lightgbm.so
[ 92%] Built target _lightgbm
[ 93%] Building CXX object CMakeFiles/lightgbm.dir/src/treelearner/gpu_tree_learner.cpp.o
[ 95%] Building CXX object CMakeFiles/lightgbm.dir/src/treelearner/serial_tree_learner.cpp.o
[ 96%] Building CXX object CMakeFiles/lightgbm.dir/src/treelearner/tree_learner.cpp.o
[ 98%] Building CXX object
CMakeFiles/lightgbm.dir/src/treelearner/voting_parallel_tree_learner.cpp.o
[100%] Linking CXX executable ../lightgbm
[100%] Built target lightgbm
```

In [18]:

```
!pip uninstall lightgbm
```

```
Uninstalling lightgbm-2.2.3:
  Would remove:
    /usr/local/lib/python3.6/dist-packages/lightgbm-2.2.3.dist-info/*
    /usr/local/lib/python3.6/dist-packages/lightgbm/*
Proceed (y/n)? y
  Successfully uninstalled lightgbm-2.2.3
```

In [19]:

```
%cd ../python-package/
!python setup.py install
```

```
/content/LightGBM/python-package
running install
INFO:LightGBM:Starting to compile the library.
INFO:LightGBM:Starting to compile with CMake.
running build
running build_py
INFO:root:Generating grammar tables from /usr/lib/python3.6/lib2to3/Grammar.txt
INFO:root:Generating grammar tables from /usr/lib/python3.6/lib2to3/PatternGrammar.txt
creating build
creating build/lib
creating build/lib/lightgbm
copying lightgbm/compat.py -> build/lib/lightgbm
copying lightgbm/callback.py -> build/lib/lightgbm
```

```
copying lightgbm/callback.py -> build/lib/lightgbm
copying lightgbm/sklearn.py -> build/lib/lightgbm
copying lightgbm/plotting.py -> build/lib/lightgbm
copying lightgbm/__init__.py -> build/lib/lightgbm
copying lightgbm/basic.py -> build/lib/lightgbm
copying lightgbm/libpath.py -> build/lib/lightgbm
copying lightgbm/engine.py -> build/lib/lightgbm
running egg_info
creating lightgbm.egg-info
writing lightgbm.egg-info/PKG-INFO
writing dependency_links to lightgbm.egg-info/dependency_links.txt
writing requirements to lightgbm.egg-info/requires.txt
writing top-level names to lightgbm.egg-info/top_level.txt
writing manifest file 'lightgbm.egg-info/SOURCES.txt'
reading manifest template 'MANIFEST.in'
no previously-included directories found matching 'build'
warning: no files found matching '*.txt'
warning: no files found matching '*.so' under directory 'lightgbm'
warning: no files found matching '*.dll' under directory 'compile/Release'
warning: no files found matching '*' under directory 'compile/compute'
warning: no files found matching '*.dll' under directory 'compile/windows/x64/DLL'
warning: no previously-included files matching '*.py[co]' found anywhere in distribution
writing manifest file 'lightgbm.egg-info/SOURCES.txt'
copying lightgbm/VERSION.txt -> build/lib/lightgbm
running install_lib
creating /usr/local/lib/python3.6/dist-packages/lightgbm
copying build/lib/lightgbm/compat.py -> /usr/local/lib/python3.6/dist-packages/lightgbm
copying build/lib/lightgbm/callback.py -> /usr/local/lib/python3.6/dist-packages/lightgbm
copying build/lib/lightgbm/sklearn.py -> /usr/local/lib/python3.6/dist-packages/lightgbm
copying build/lib/lightgbm/plotting.py -> /usr/local/lib/python3.6/dist-packages/lightgbm
copying build/lib/lightgbm/__init__.py -> /usr/local/lib/python3.6/dist-packages/lightgbm
copying build/lib/lightgbm/basic.py -> /usr/local/lib/python3.6/dist-packages/lightgbm
copying build/lib/lightgbm/VERSION.txt -> /usr/local/lib/python3.6/dist-packages/lightgbm
copying build/lib/lightgbm/libpath.py -> /usr/local/lib/python3.6/dist-packages/lightgbm
copying build/lib/lightgbm/engine.py -> /usr/local/lib/python3.6/dist-packages/lightgbm
INFO:LightGBM:Installing lib_lightgbm from: ['../lib_lightgbm.so', 'compile/lib_lightgbm.so']
copying ../lib_lightgbm.so -> /usr/local/lib/python3.6/dist-packages/lightgbm
byte-compiling /usr/local/lib/python3.6/dist-packages/lightgbm/compat.py to compat.cpython-36.pyc
byte-compiling /usr/local/lib/python3.6/dist-packages/lightgbm/callback.py to callback.cpython-36.
pyc
byte-compiling /usr/local/lib/python3.6/dist-packages/lightgbm/sklearn.py to sklearn.cpython-36.py
c
byte-compiling /usr/local/lib/python3.6/dist-packages/lightgbm/plotting.py to plotting.cpython-36.
pyc
byte-compiling /usr/local/lib/python3.6/dist-packages/lightgbm/__init__.py to __init__.cpython-36.
pyc
byte-compiling /usr/local/lib/python3.6/dist-packages/lightgbm/basic.py to basic.cpython-36.pyc
byte-compiling /usr/local/lib/python3.6/dist-packages/lightgbm/libpath.py to libpath.cpython-36.py
c
byte-compiling /usr/local/lib/python3.6/dist-packages/lightgbm/engine.py to engine.cpython-36.pyc
running install_egg_info
Copying lightgbm.egg-info to /usr/local/lib/python3.6/dist-packages/lightgbm-3.0.0.99-py3.6.egg-in
fo
running install_scripts
```

In [34]:

```
param = {'objective' : ['multiclass'], 'boosting_type' : ['gbdt'],'learning_rate': [0.05,0.1],'num_
leaves': [10,50,100],'bagging_fraction' : [0.7],'feature_fraction' : [0.7],
             'bagging_seed' : [420],'max_depth' : [2,5,7],'device':['gpu'],'metric' : ['multi_logl
ss'],'num_class':[4]}

model_lgb = lgb.LGBMClassifier()
random_lgb = RandomizedSearchCV(model_lgb,param)
random_lgb.fit(X_train,y_train)
```

```
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
```

```
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
```

```
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
[LightGBM] [Warning] feature_fraction is set=0.7, colsample_bytree=1.0 will be ignored. Current va
lue: feature_fraction=0.7
[LightGBM] [Warning] bagging_fraction is set=0.7, subsample=1.0 will be ignored. Current value: ba
gging_fraction=0.7
```

Out[34]:

```
RandomizedSearchCV(cv=None, error_score=nan,
                   estimator=LGBMClassifier(boosting_type='gbdt',
                                            class_weight=None,
                                            colsample_bytree=1.0,
                                            importance_type='split',
                                            learning_rate=0.1, max_depth=-1,
                                            min_child_samples=20,
                                            min_child_weight=0.001,
                                            min_split_gain=0.0,
                                            n_estimators=100, n_jobs=-1,
                                            num_leaves=31, objective=None,
                                            random_state=None, reg_alpha=0.0,
                                            reg_lambda=0.0, s...
                   param_distributions={'bagging_fraction': [0.7],
                                        'bagging_seed': [420],
                                        'boosting_type': ['gbdt'],
                                        'device': ['gpu'],
                                        'feature_fraction': [0.7],
                                        'learning_rate': [0.05, 0.1],
                                        'max_depth': [2, 5, 7],
                                        'metric': ['multi_logloss'],
                                        'num_class': [4],
                                        'num_leaves': [10, 50, 100],
                                        'objective': ['multiclass']},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=False, scoring=None, verbose=0)
```

In [35]:

```
random_lgb.best_params_
```

Out[35]:

```
{'bagging_fraction': 0.7,
 'bagging_seed': 420,
 'boosting_type': 'gbdt',
 'device': 'gpu',
 'feature_fraction': 0.7,
 'learning_rate': 0.1,
 'max_depth': 7,
 'metric': 'multi_logloss',
 'num_class': 4,
 'num_leaves': 10,
 'objective': 'multiclass'}
```

In [17]:

```
X_train.head()
```

Out[17]:

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp
328831	1	DA	352.382812	0	0.407010	0.432705	0.433308	0.461310	0.574949	0.516980	0.457925	0.39848

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp
2981344	6	DA	340.722656	0	0.404421	0.432482	0.429756	0.462451	0.572485	0.515401	0.452219	0.39429
650858	2	DA	131.117188	1	0.404986	0.432844	0.433020	0.466871	0.576092	0.519862	0.458077	0.39482
915303	2	SS	272.736023	0	0.406429	0.433559	0.426334	0.463166	0.573996	0.519485	0.453143	0.39135
4711282	13	SS	147.885010	0	0.459246	0.467483	0.461418	0.475050	0.574385	0.525673	0.466746	0.44520

In [20]:

```
lgbtrain = lgb.Dataset(X_train, y_train)
lgbtest = lgb.Dataset(X_test, y_test)
```

In [55]:

```
params = {'bagging_fraction': 0.7,
 'bagging_seed': 420,
 'boosting_type': 'gbdt',
 'feature_fraction': 0.7,
 'learning_rate': 0.1,
 'max_depth': 7,
 'metric': 'multi_error',
 'num_class': 4,
 'num_leaves': 10,
 'objective': 'multiclass'}
model_lgb = lgb.train(params, lgbtrain, 1000, valid_sets=[lgbtest], early_stopping_rounds=50, verbo
se_eval=100)
```

```
Training until validation scores don't improve for 50 rounds.
[100] valid_0's multi_error: 0.038167
[200] valid_0's multi_error: 0.0268068
[300] valid_0's multi_error: 0.0193696
[400] valid_0's multi_error: 0.0139848
[500] valid_0's multi_error: 0.0115246
[600] valid_0's multi_error: 0.457553
Early stopping, best iteration is:
[582] valid_0's multi_error: 0.00975362
```

In [56]:

```
model_lgb.best_iteration
```

Out[56]:

```
582
```

In [51]:

```
predicted_lgb = model_lgb.predict(X_test, num_iteration= model_lgb.best_iteration)
print('Log loss',round(log_loss(y_test.to_numpy(),predicted_lgb),8))
```

```
Log loss 0.01693478
```

In [61]:

```
joblib.dump(model_lgb, 'lightgbm.pkl')
```

Out[61]:

```
['lightgbm.pkl']
```

In [58]:

```
predicted_lgb = model_lgb.predict(df_test_final, num_iteration= model_lgb.best_iteration)
# predicted_dt = clf_dt.predict_proba(df_test_final)
```

In [59]:

```
submission = pd.DataFrame(np.concatenate((np.arange(len(df_test_final))[:, np.newaxis],
predicted_lgb), axis=1), columns=['id', 'A', 'B', 'C', 'D'])
submission['id'] = df_test_id.astype(int)
submission.to_csv("Submission.csv", index=False)
```

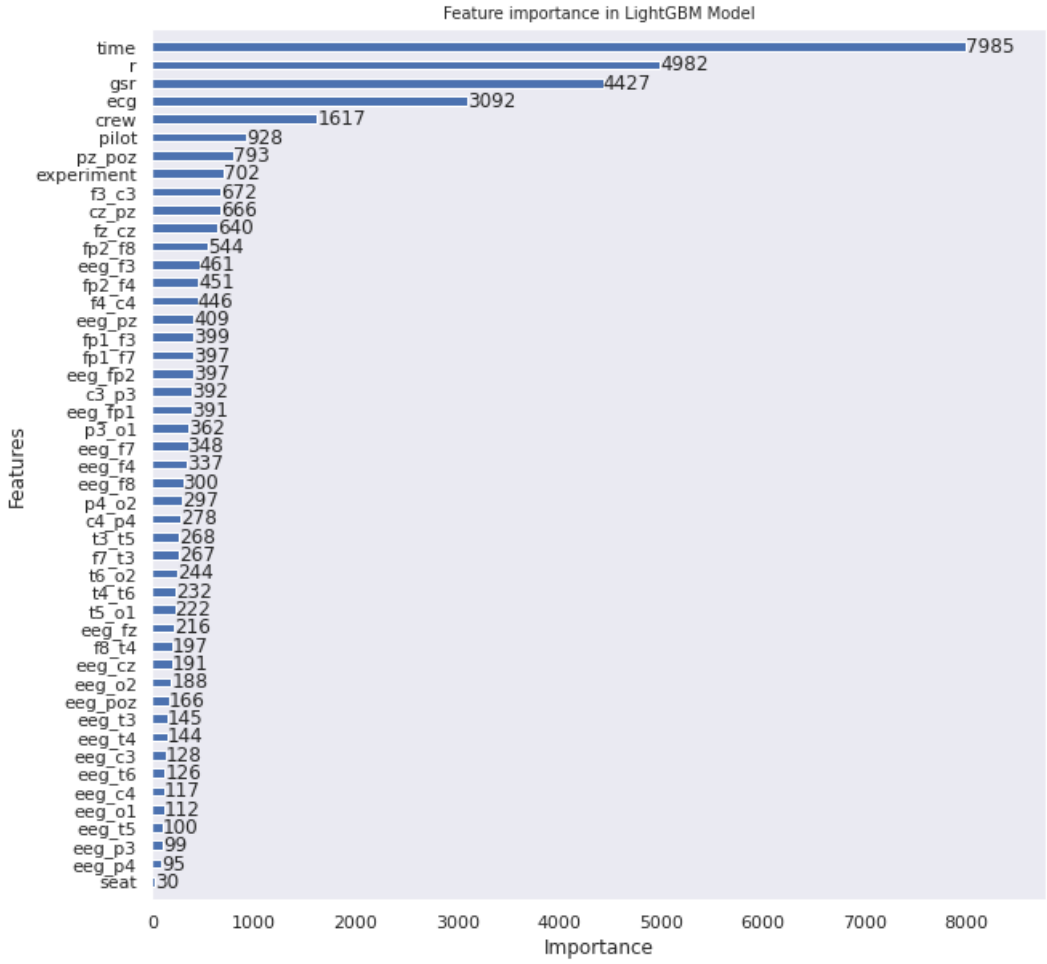
In [62]:

```
!kaggle competitions submit -c reducing-commercial-aviation-fatalities -f Submission.csv -m "lgb"
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run '
chmod 600 /content/kaggle.json'
Warning: Looks like you're using an outdated API Version, please consider updating (server 1.5.6 /
client 1.5.4)
100% 1.58G/1.58G [00:18<00:00, 90.7MB/s]
Successfully submitted to Reducing Commercial Aviation Fatalities
```

In [62]:

```
sns.set(style="darkgrid")
fig,ax = plt.subplots(figsize=(10,10))
lgb.plot_importance(model_lgb, height=0.5, ax=ax)
ax.grid(False)
plt.ylabel('Features', size=12)
plt.xlabel('Importance', size=12)
plt.title("Feature importance in LightGBM Model", fontsize=10)
plt.show()
```



Results

In [52]:

```
table_result = PrettyTable()
table_result.field_names = ['Model','Log loss']
table_result.add_row(['Decision Tree','0.037880678586169'])
table_result.add_row(['Random Forest','0.05680608782397054'])
table_result.add_row(['Logistic Regression','0.29901234963328643'])
```

```
table_result.add_row(['Logistic Regression',' 0.29901234963328643'])
table_result.add_row(['LightGBM',' 0.01693478'])
```

In [54]:

```
print(table_result)
```

```
+---------------------+---------------------+
|        Model        |       Log loss      |
+---------------------+---------------------+
|    Decision Tree    |  0.037880678586169  |
|    Random Forest    | 0.05680608782397054 |
| Logistic Regression | 0.29901234963328643 |
|       LightGBM      |      0.01693478     |
+---------------------+---------------------+
```

Now we know from our EDA that we have huge class imbalance so we will try a technique called SMOTE and check if it is good to use in our case or not

One approach to addressing imbalanced datasets is to oversample the minority class. The simplest approach involves duplicating examples in the minority class, although these examples don't add any new information to the model. Instead, new examples can be synthesized from the existing examples. This is a type of data augmentation for the minority class and is referred to as the Synthetic Minority Oversampling Technique, or SMOTE

In [19]:

```
col_names = df_train.columns
y = df_train['event']
df_train,y = SMOTE().fit_resample(df_train,y.ravel())
df_train = pd.DataFrame(df_train)
```

In [20]:

```
df_train.columns = col_names
train, test = train_test_split(df_train, test_size=0.2, random_state=42, shuffle=True)
```

In [21]:

```
X_train = train.loc[:, df_train.columns != 'event']
y_train = train.event

X_test = test.loc[:, df_train.columns != 'event']
y_test = test.event

print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
(9116188, 47) (9116188,)
(2279048, 47) (2279048,)
```

Random Forest

In [29]:

```
clf_rf = RandomForestClassifier(criterion='entropy',max_depth=100,n_estimators=10,random_state=100
)
clf_rf = clf_rf.fit(X_train,y_train)
```

In [30]:

```
predicted_rf = clf_rf.predict_proba(X_test)
loss_rf = log_loss(y_test,predicted_rf)
print('Log loss = ',loss_rf)
```

```
Log loss =  0.026754095057236256
```

Logistic Regression

In [31]:

```
clf_lg = LogisticRegression(C=10,multi_class='ovr',penalty='l2',random_state=100)
clf_lg = clf_lg.fit(X_train,y_train)
```

In [32]:

```
predicted_lg = clf_lg.predict_proba(X_test)
loss_lg = log_loss(y_test,predicted_lg)
print('Log loss = ',loss_lg)
```

```
Log loss =  0.6363916769604432
```

Decision Tree

In [27]:

```
clf_dt = DecisionTreeClassifier(criterion='entropy',max_depth=100,max_leaf_nodes=200,random_state=
100)
clf_dt = clf_dt.fit(X_train,y_train)
```

In [28]:

```
predicted_dt = clf_dt.predict_proba(X_test)
loss_dt = log_loss(y_test,predicted_dt)
print('Log loss = ',loss_dt)
```

```
Log loss =  0.04818923787998907
```

LightGBM

In [22]:

```
lgbtrain = lgb.Dataset(X_train, y_train)
lgbtest = lgb.Dataset(X_test, y_test)
```

In [24]:

```
params = {'bagging_fraction': 0.7,
 'bagging_seed': 420,
 'boosting_type': 'gbdt',
 'feature_fraction': 0.7,
 'learning_rate': 0.1,
 'max_depth': 7,
 'metric': 'multi_logloss',
 'num_class': 4,
 'num_leaves': 10,
 'objective': 'multiclass'}
model_lgb = lgb.train(params, lgbtrain, 1000, valid_sets=[lgbtest], early_stopping_rounds=50, verbo
se_eval=100)
```

```
Training until validation scores don't improve for 50 rounds.
[100] valid_0's multi_logloss: 0.131484
[200] valid_0's multi_logloss: 0.0912677
[300] valid_0's multi_logloss: 0.0753155
[400] valid_0's multi_logloss: 0.0639219
[500] valid_0's multi_logloss: 0.0559704
[600] valid_0's multi_logloss: 0.0494358
[700] valid_0's multi_logloss: 0.0442652
[800] valid_0's multi_logloss: 0.0397725
[900] valid_0's multi_logloss: 0.0363426
[1000] valid_0's multi_logloss: 0.033711
Did not meet early stopping. Best iteration is:
[1000] valid_0's multi_logloss: 0.033711
```

In [25]:

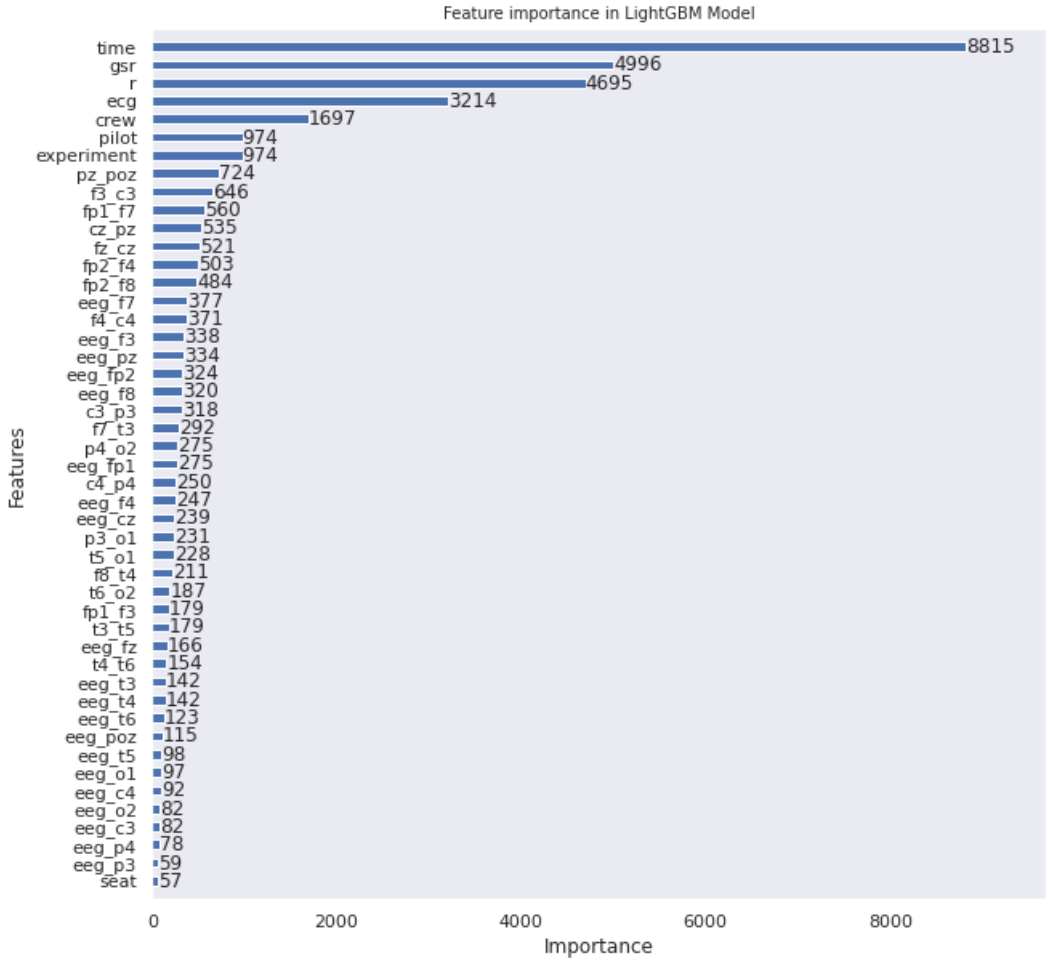
In [25]:

```
predicted_lgb = model_lgb.predict(X_test, num_iteration= model_lgb.best_iteration)
print('Log loss',round(log_loss(y_test.to_numpy(),predicted_lgb),8))
```

```
Log loss 0.03371104
```

In [26]:

```
sns.set(style="darkgrid")
fig,ax = plt.subplots(figsize=(10,10))
lgb.plot_importance(model_lgb, height=0.5, ax=ax)
ax.grid(False)
plt.ylabel('Features', size=12)
plt.xlabel('Importance', size=12)
plt.title("Feature importance in LightGBM Model", fontsize=10)
plt.show()
```



Results

In [33]:

```
table_result = PrettyTable()
table_result.field_names = ['Model','Log loss']
table_result.add_row(['Decision Tree','0.04818923787998907'])
table_result.add_row(['Random Forest','0.026754095057236256'])
table_result.add_row(['Logistic Regression','0.6363916769604432'])
table_result.add_row(['LightGBM','0.03371104'])
print(table_result)
```

```
+---------------------+----------------------+
|        Model        |       Log loss       |
+---------------------+----------------------+
|    Decision Tree    | 0.04818923787998907  |
|    Random Forest    | 0.026754095057236256 |
| Logistic Regression |  0.6363916769604432  |
|       LightGBM      |      0.03371104      |
```

+----------------------+----------------------+

References:

- Haberman Dataset Assignment
- Linear SVM Assignment
- Reference 1
- Reference 2
- Reference 3
- Reference 4
- Reference 5
- Reference 6
- Reference 7
- Reference 8
- Reference 9
- Reference 10
- Reference 11
- Reference 12
- Reference 13
- Reference 14
- Reference 15
- Reference 16
- Reference 17
- Reference 18
- Reference 19