Cairo University

Features extraction and analysis of ECG signal

Team 7

Task 3

Report by:

- Dina Mostafa Sec: 1 B.N: 30
- Sara Tarek Sec: 1 B.N: 40
- Ahmed Ashraf Sec: 1 B.N: 3
- Mohamed Salah Sec: 2 B.N: 19
- Yassmin Yasser Sec: 2 B.N: 52

Supervised By:

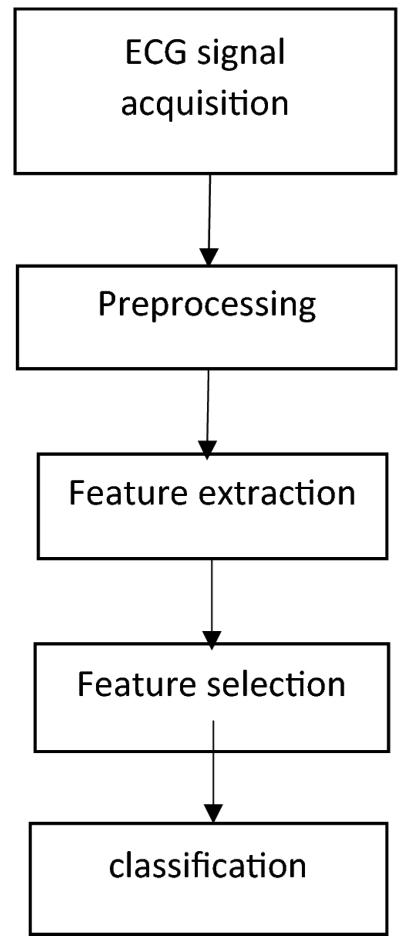
Dr. Ahmed Morsy & Dr.Eman Ayman

TA: eng.Abdelrahman Hesham

Description:

To date 84 kind of sleep disorders have been discovered, where insomnia, sleep apnea, narcolepsy, and restless leg syndrome are the most common sleep disorders.

ECG recording is one of the simpler and efficient technology in sleep disorders detection, variations in RR intervals (beat to beat heart rate) of ECG signals is associated with sleep apnea events.



The processes for the signal analysis:

(a)Data acquisition
(b)Data pre-processing
(c)Feature extraction
(d)Feature selection
(e)Model training and classification
(f)Performance evaluation

1.Subject:

the database of ECG signals consists of 35 record, containing a single ECG signal digitized at 100 Hz with 16-bit resolution, continuously for approximately 8 hours (individual recordings vary in length from less than 7 hours to nearly 10 hours). Each recording includes a set of reference annotations, one for each minute of the recording (epoch), indicate the presence or absence of apnea during that minute. These reference annotations were made by human experts on the basis of simultaneously recorded respiration signals.

2. Feature extraction:

The basic goal of feature extraction is for dimensionality reduction and data compaction and for getting the best accuracy by finding the highly correlated features with the output for good accuracy of the model .

1) Time domain feature extraction:

- At first we realized that in the paper was said that each .apn file is containing the annotations that says if it is apnea or not “A” for apnea and “N” for not apnea

- the files of the .dat record is a complete record and each record has its own .apn file and each annotation in it indicates for the result of 1 minute

- so the first step we read the .dat files and then read the .apn file and for the size of the .apn file we divided the file of the .dat with respect to it in example if we have 30 annotation so we divide the .dat for 30 subarray

- and then we saved this records in 2d array then each subarray is representation for 1 minute and then looping on all records [“a”,”b”,”c”] which represents 35 record and the result of the annotations was with size 17045 and respecting to that we had 17045 subarray

```python
appnea=[]
data=[]
ecg=[['a','b','c'],[20,5,10]]
for iterate in range(0,3):
    x='0'
    size=ecg[1][iterate]
    for i in range(1,size+1):
        if(i==10):
            x=''
        signal= wfdb.rdrecord(record_name=("apnea-ecg-database-1.0.0/"+ecg[0][iterate]+ x + str(i) ) ).p_signal
        signal = signal.reshape(len(signal))
        annotation = wfdb.rdann(("apnea-ecg-database-1.0.0/"+ecg[0][iterate]+ x + str(i) ), extension="apn")
        recordSample=int(len(signal)/len(annotation.symbol))
        for j in range(0,recordSample*len(annotation.symbol), recordSample):
            data.append(signal[j:j+recordSample])
        for element in annotation.symbol:
            appnea.append(element)

# appnea=np.array(appnea).ravel()
print(len(appnea))
print(len(data))
```

Python

```
17045
17045
```

- then we got the time and frequency domain features and like mean ,median ,RMS ,Var ,Crest factor ,skew,kurtosis,etc… and that with help of this link https://matteogambera.medium.com/how-to-extract-features-from-signals-15e7db225c15

```python
FEATURES = ['MIN','MAX','MEAN','RMS','VAR','STD','POWER','PEAK','P2P','CREST FACTOR','SKEW','KURTOSIS',
            'MAX_f','SUM_f','MEAN_f','VAR_f','PEAK_f','SKEW_f','KURTOSIS_f',"Appnea"]

Min=[];Max=[];Mean=[];Rms=[];Var=[];Std=[];Power=[];Peak=[];Skew=[];Kurtosis=[];P2p=[];CrestFactor=[];
FormFactor=[]; PulseIndicator=[];
Max_f=[];Sum_f=[];Mean_f=[];Var_f=[];Peak_f=[];Skew_f=[];Kurtosis_f=[]
```

```python
i=0
for signal in data:
    print(len(signal))
    i=i+1
    X = signal
    ## TIME DOMAIN ##

    Min.append(np.min(X))
    Max.append(np.max(X))
    Mean.append(np.mean(X))
    Rms.append(np.sqrt(np.mean(X**2)))
    Var.append(np.var(X))
    Std.append(np.std(X))
    Power.append(np.mean(X**2))
    Peak.append(np.max(np.abs(X)))
    P2p.append(np.ptp(X))
    CrestFactor.append(np.max(np.abs(X))/np.sqrt(np.mean(X**2)))
    Skew.append(stats.skew(X))
    Kurtosis.append(stats.kurtosis(X))
    FormFactor.append(np.sqrt(np.mean(X**2))/np.mean(X))
    PulseIndicator.append(np.max(np.abs(X))/np.mean(X))
    ## FREQ DOMAIN ##
    ft = fft(X)
    S = np.abs(ft**2)/len(X)
    Max_f.append(np.max(S))
    Sum_f.append(np.sum(S))
    Mean_f.append(np.mean(S))
    Var_f.append(np.var(S))
    
    Peak_f.append(np.max(np.abs(S)))
    Skew_f.append(stats.skew(X))
    Kurtosis_f.append(stats.kurtosis(X))
```

- then we got this features for all the subarrays and then created data frame with this features and data

- this data frame was represented in form of rows for the features so we did transpose for it

```
df_features = pd.DataFrame(index = [FEATURES],
                           data = [Min,Max,Mean,Rms,Var,Std,Power,Peak,P2p,CrestFactor,Skew,Kurtosis,
                                   Max_f,Sum_f,Mean_f,Var_f,Peak_f,Skew_f,Kurtosis_f,appnea])
df_features=df_features.transpose()
df_features
```

Python

	MIN	MAX	MEAN	RMS	VAR	STD	POWER	PEAK	P2P	CREST FACTOR	SKEW	KURTOSIS	MAX_f	SUM_f	MEAN_f
0	-0.885	1.75	-0.000796	0.254363	0.0647	0.254362	0.064701	1.75	2.635	6.87993	2.827631	16.328233	2.033477	391.244275	0.064701
1	-0.9	1.74	0.000511	0.26962	0.072695	0.269619	0.072695	1.74	2.64	6.453531	2.689177	14.858693	4.027249	439.5858	0.072695
2	-1.035	1.79	0.000175	0.273829	0.074982	0.273829	0.074982	1.79	2.825	6.536922	2.467887	13.894459	1.84386	453.41875	0.074982
3	-1.17	1.845	-0.0006	0.280689	0.078786	0.280689	0.078787	1.845	3.015	6.573101	1.916639	11.294911	2.014	476.42235	0.078787
4	-1.035	1.7	0.000928	0.266043	0.070778	0.266041	0.070779	1.7	2.735	6.389942	2.119219	11.757525	4.485164	428.00025	0.070779
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17040	-0.845	1.255	0.003319	0.176684	0.031206	0.176653	0.031217	1.255	2.1	7.103086	2.526152	12.966127	2.15035	187.209275	0.031217
17041	-0.31	1.02	0.002804	0.139519	0.019458	0.139491	0.019466	1.02	1.33	7.310829	4.481259	24.558055	1.017345	116.735025	0.019466
17042	-0.385	1.02	0.002916	0.145594	0.021189	0.145565	0.021198	1.02	1.405	7.005777	3.980178	21.699035	2.611107	127.122325	0.021198
17043	-0.41	1.145	0.003498	0.152217	0.023158	0.152177	0.02317	1.145	1.555	7.522144	3.799732	20.161336	1.506311	138.951025	0.02317
17044	-2.98	2.705	-0.006169	0.189315	0.035802	0.189214	0.03584	2.98	5.685	15.740998	2.12957	36.290941	1.160465	214.932475	0.03584

17045 rows × 20 columns

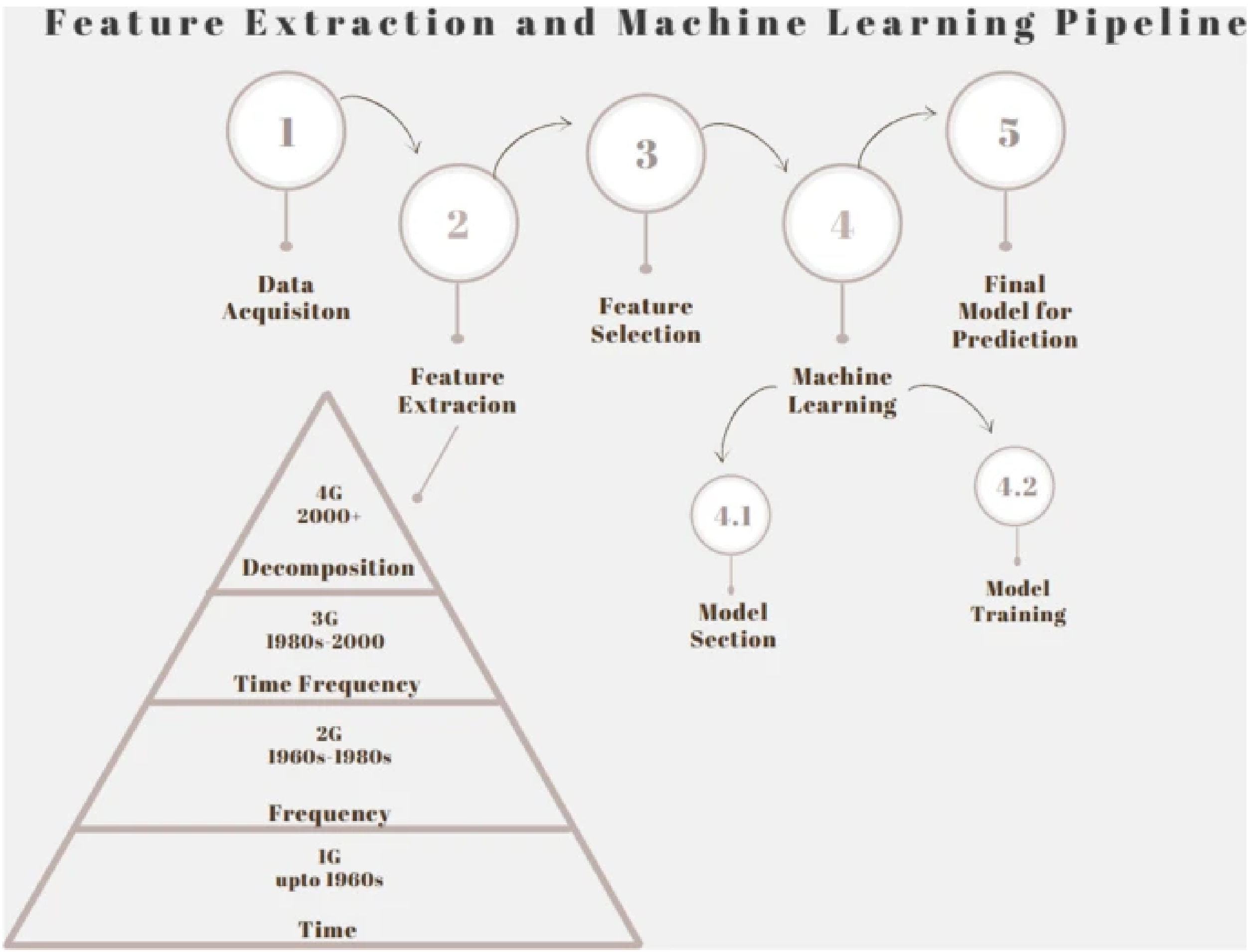
- finally we created a csv file which is named “main features time & frequency domain.csv“ for all this feature

```
df_features.to_csv("isAppneaFeatures.csv")
```

- but when we tried to do the feature selection and dimensionality reduction we realized that this data after getting the features some of them were missed

- we dropped the rows of the missed data as we realized that they weren’t too much and then saved them in “cleaned features.csv” file

- finally after this we started the feature selection phase



3. Feature selection:

After extract features from ECG signal, it’s time to select features from dataset there are different techniques to select features they are :

- Pearson Correlation
- Information Gain (entropy or Gini impurity)
- Chi2

In our task we use Pearson Correlation, but before it use it we must clean data from null Values by deleting rows which has null values

```

# check for null values
has_null = df.isnull().any(axis=1)

# get index of rows with null values
null_index = df.index[has_null]

print(f"DataFrame has {df.isnull().sum().sum()} null values")
```

after dropping this rows, we check if there are constant columns and delete this features

```python
var_thres = VarianceThreshold(threshold=0)
var_thres.fit(X_data)
```

[227] Python

```
VarianceThreshold
VarianceThreshold(threshold=0)
```

```python
var_thres.get_support()
```

[228] Python

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True])
```

```python
### Finding non constant features
sum(var_thres.get_support())
```

[229] Python

```
19
```

```python
# Lets Find non-constant features
len(X_data.columns[var_thres.get_support()])
```

[230] Python

```
19
```

```python
constant_columns = [column for column in X_data.columns
                    if column not in X_data.columns[var_thres.get_support()]]

print(len(constant_columns))
```

[231] Python

```
0
```

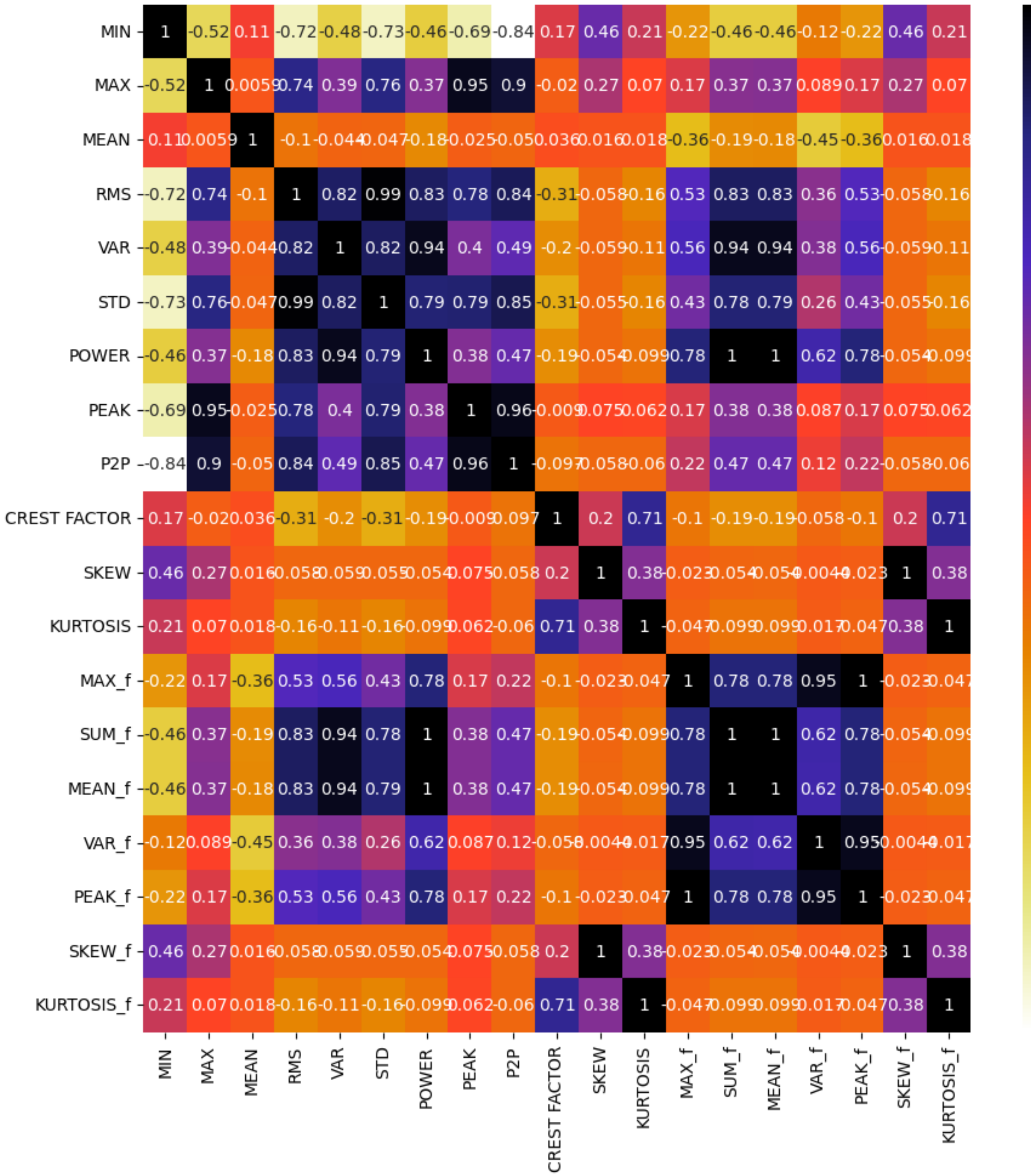
```python
for column in constant_columns:
    print(column)
```

[232] Python

```python
X_data.drop(constant_columns,axis=1)
```

[233] Python

now we get correlation between each feature and other features so we get this result



by using correlation function to get features max features that has high correlation according to threshold that given to function and drop this features from my data set

```
# with the following function we can select highly correlated features

def correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            # we are interested in absolute coeff value
            if abs(corr_matrix.iloc[i, j]) > threshold:
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr

[235]
```

```
corr_features = correlation(X_data, 0.9)
len(set(corr_features))

[236]
```

```
... 10
```

```
corr_features

[237]
```

```
... {'KURTOSIS_f',
 'MEAN_f',
 'P2P',
 'PEAK',
 'PEAK_f',
 'POWER',
 'SKEW_f',
 'STD',
 'SUM_f',
 'VAR_f'}
```

```
X_data_from_Corrolation = X_data.drop(corr_features,axis=1)
X_data_from_Corrolation

[238]
```

so my data is ready for next step for Feature reduction.

4. Feature reduction:

The purpose of using feature reduction is to reduce the number of features (or variables) that the computer must process to perform its function.

PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate, the second greatest variance on the second coordinate, and so on.

To find out how PCA transformation could improve the classification accuracy, we perform apnea classification using PC as a features.

SVM with RBF kernel shows the best classification accuracy on the Cross Validation and Random Sampling model, while kNN shows the best performance on test on Train Data.

—---

Explain

1.First it was in data problems, it was in data loss.

2. Secondly, we defined the number of row and column and determined their number, then we read the data before the reduction and after the reduction.

3.it was before the reduction of 17 thousand data and the number of features 10, then we did some pre-processing to prepare the data.

4.then we applied the pca which is the technique Which we used to reduce features, then we printed the shape again so it became 5 features.

6.finally, we converted the data to a data frame to keep it in csv, then we printed all the new feature.

MIN	MAX	MEAN	RMS	VAR	STD	POWER	PEAK	P2P	CREST FAC	SKEW	KURTOSIS	MAX_f	SUM_f	MEAN_f	VAR_f	PEAK_f	SKEW_f	KURTOSIS_	Appnea
-0.885	1.75	-0.0008	0.254363	0.0647	0.254362	0.064701	1.75	2.635	6.87993	2.827631	16.32823	2.033477	391.2443	0.064701	0.019939	2.033477	2.827631	16.32823	0
-0.9	1.74	0.000511	0.26962	0.072695	0.269619	0.072695	1.74	2.64	6.453531	2.689177	14.85869	4.027249	439.5858	0.072695	0.026968	4.027249	2.689177	14.85869	0
-1.035	1.79	0.000175	0.273829	0.074982	0.273829	0.074982	1.79	2.825	6.536922	2.467887	13.89446	1.84386	453.4187	0.074982	0.021838	1.84386	2.467887	13.89446	0
-1.17	1.845	-0.0006	0.280689	0.078786	0.280689	0.078787	1.845	3.015	6.573101	1.916639	11.29491	2.014	476.4223	0.078787	0.024715	2.014	1.916639	11.29491	0
-1.035	1.7	0.000928	0.266043	0.070778	0.266041	0.070779	1.7	2.735	6.389942	2.119219	11.75753	4.485164	428.0002	0.070779	0.027148	4.485164	2.119219	11.75753	0
-0.895	1.795	0.001346	0.259864	0.067527	0.25986	0.067529	1.795	2.69	6.907467	2.577358	14.47294	1.905401	408.3487	0.067529	0.019019	1.905401	2.577358	14.47294	0
-0.935	1.795	-2.89E-05	0.266646	0.0711	0.266646	0.0711	1.795	2.73	6.731783	2.542474	14.21189	1.915266	429.9408	0.0711	0.023024	1.915266	2.542474	14.21189	0
-1.005	1.88	9.92E-06	0.281667	0.079336	0.281667	0.079336	1.88	2.885	6.674552	2.121365	10.99851	2.029081	479.746	0.079336	0.023532	2.029081	2.121365	10.99851	0
-1.15	1.785	0.001294	0.259157	0.067161	0.259154	0.067162	1.785	2.935	6.887716	1.919977	11.08926	1.17071	406.1308	0.067162	0.015458	1.17071	1.919977	11.08926	0
-0.86	1.77	2.65E-05	0.257661	0.066389	0.257661	0.066389	1.77	2.63	6.869489	2.454217	14.08002	1.4208	401.4557	0.066389	0.016143	1.4208	2.454217	14.08002	0
-0.915	1.745	0.000367	0.255858	0.065463	0.255858	0.065463	1.745	2.66	6.82018	2.572547	14.80686	1.661813	395.8577	0.065463	0.017173	1.661813	2.572547	14.80686	0
-0.905	1.82	-0.00027	0.245541	0.06029	0.245541	0.06029	1.82	2.725	7.412213	2.64505	15.49808	1.275353	364.5751	0.06029	0.015418	1.275353	2.64505	15.49808	0
-0.93	1.875	-0.00033	0.251014	0.063008	0.251014	0.063008	1.875	2.805	7.469708	2.753099	16.05029	2.107513	381.009	0.063008	0.017869	2.107513	2.753099	16.05029	0
-0.91	1.925	0.000208	0.249942	0.062471	0.249942	0.062471	1.925	2.835	7.701776	2.940079	18.12297	1.276894	377.7632	0.062471	0.01463	1.276894	2.940079	18.12297	1
-0.985	1.975	0.000988	0.249593	0.062296	0.249591	0.062297	1.975	2.96	7.912887	2.818327	17.20043	1.446816	376.7075	0.062297	0.014317	1.446816	2.818327	17.20043	1
-0.94	1.92	-0.00074	0.249538	0.062269	0.249537	0.062269	1.92	2.86	7.694218	2.905401	17.68854	1.158107	376.542	0.062269	0.01464	1.158107	2.905401	17.68854	1
-1.065	1.95	0.000227	0.249285	0.062143	0.249285	0.062143	1.95	3.015	7.822359	2.866377	17.72739	2.238882	375.78	0.062143	0.016254	2.238882	2.866377	17.72739	1
-0.94	1.89	0.000196	0.254235	0.064635	0.254235	0.064635	1.89	2.83	7.434069	2.914354	17.64471	2.326944	390.8502	0.064635	0.016705	2.326944	2.914354	17.64471	1
-0.93	1.925	-0.00031	0.250701	0.062851	0.2507	0.062851	1.925	2.855	7.678481	2.967045	18.21776	1.71287	380.0588	0.062851	0.015472	1.71287	2.967045	18.21776	1
-0.975	1.915	0.000395	0.253825	0.064427	0.253825	0.064427	1.915	2.89	7.544573	2.978066	17.94372	1.617208	389.5903	0.064427	0.01547	1.617208	2.978066	17.94372	1
-0.98	1.965	0.000742	0.248532	0.061767	0.248531	0.061768	1.965	2.945	7.906434	3.025025	18.73384	1.23155	373.5114	0.061768	0.014636	1.23155	3.025025	18.73384	1
-0.97	1.96	0.000602	0.250835	0.062918	0.250834	0.062918	1.96	2.93	7.813898	3.008708	18.80404	1.523072	380.4666	0.062918	0.015638	1.523072	3.008708	18.80404	1
-0.965	1.955	0.000402	0.245937	0.060485	0.245937	0.060485	1.955	2.92	7.949179	2.949583	18.18591	1.367557	365.7539	0.060485	0.01407	1.367557	2.949583	18.18591	1
-0.96	2.085	0.000356	0.252077	0.063543	0.252077	0.063543	2.085	3.045	8.271274	3.042147	18.98825	1.07911	384.2442	0.063543	0.015031	1.07911	3.042147	18.98825	1
-0.905	1.84	-0.00021	0.24257	0.05884	0.24257	0.05884	1.84	2.745	7.585427	2.997139	18.78572	1.127883	355.8078	0.05884	0.013163	1.127883	2.997139	18.78572	1
-1.01	1.905	0.000937	0.252673	0.063843	0.252671	0.063844	1.905	2.915	7.539389	2.954319	18.19005	1.333852	386.0625	0.063844	0.015216	1.333852	2.954319	18.19005	1
-0.94	1.915	0.00037	0.240715	0.057943	0.240715	0.057944	1.915	2.855	7.955472	3.040962	19.12624	1.128432	350.3851	0.057944	0.012742	1.128432	3.040962	19.12624	1
-1.035	1.95	0.000473	0.248407	0.061706	0.248407	0.061706	1.95	2.985	7.850005	3.029071	18.97427	1.766386	373.1378	0.061706	0.014837	1.766386	3.029071	18.97427	1
-0.93	2.025	0.000504	0.246776	0.060898	0.246776	0.060899	2.025	2.955	8.205811	2.93537	18.33063	1.47059	368.2536	0.060899	0.013943	1.47059	2.93537	18.33063	1
-0.93	1.975	0.000492	0.24595	0.060491	0.24595	0.060491	1.975	2.905	8.030083	3.091407	19.3577	2.115254	365.7919	0.060491	0.01721	2.115254	3.091407	19.3577	1
-0.915	2.15	-0.00012	0.248123	0.061565	0.248123	0.061565	2.15	3.065	8.66507	3.018788	18.93058	1.255641	372.2826	0.061565	0.013853	1.255641	3.018788	18.93058	1

feature reduction

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler , LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, ConfusionMatrixDisplay
from scipy.stats import randint
from sklearn.decomposition import PCA
```

[7]

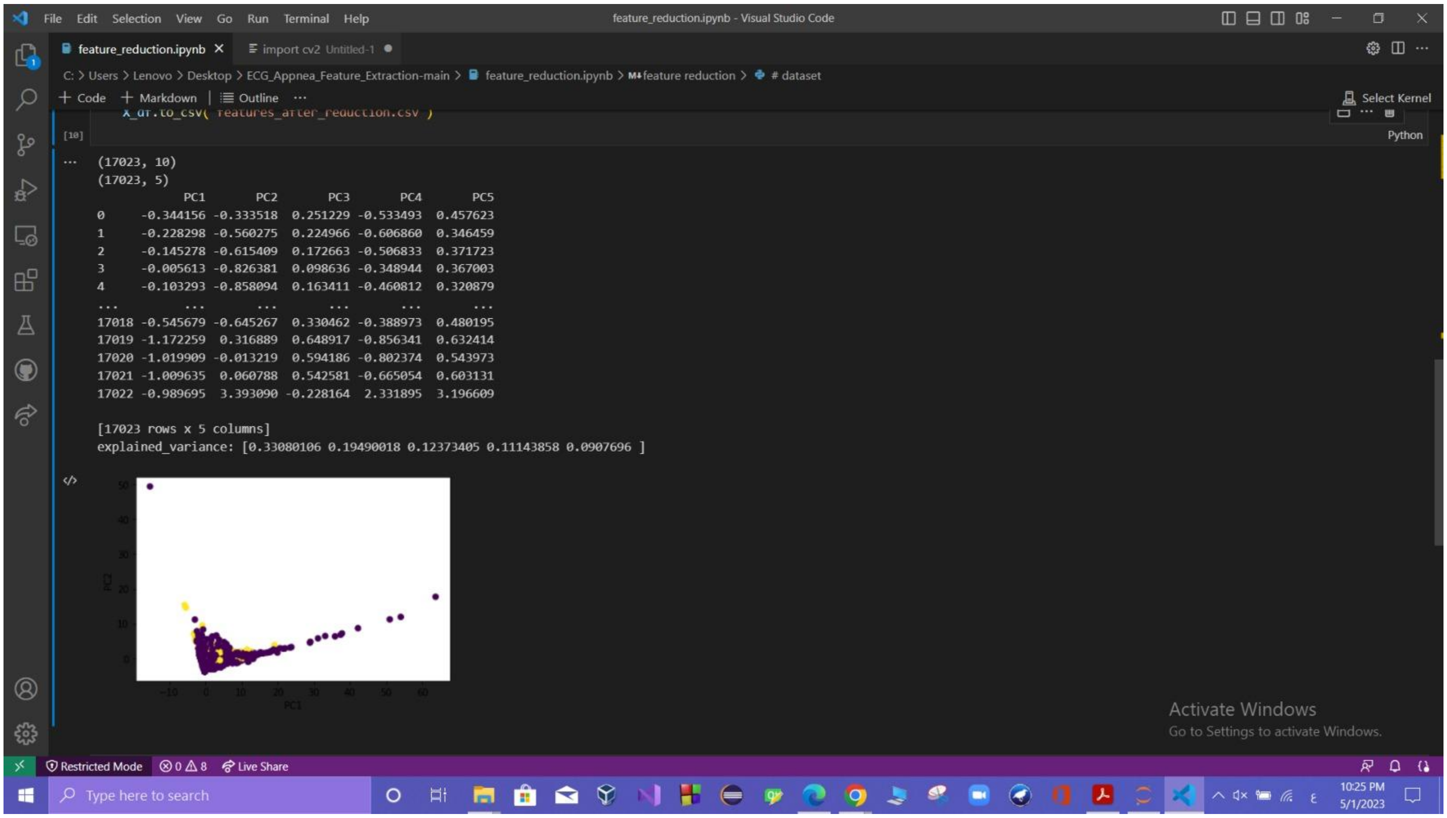
```python
# dataset
dataset = pd.read_csv('cleaned data after select features.csv')
# distributing the dataset into two components X and Y
X = dataset.iloc[:, 0:17000].values
y = dataset.iloc[:, 9].values
print(X.shape)
#performing preprocessing part
sc = StandardScaler()
X = sc.fit_transform(X)
le = LabelEncoder()
y = le.fit_transform(y)
# Applying PCA function on training
# and testing set of X component
pca = PCA(n_components = 5)
X = pca.fit_transform(X)
print(X.shape)
# Convert numpy arrays to pandas DataFrame
X_df = pd.DataFrame(X, columns=['PC1', 'PC2','PC3','PC4','PC5'])
# Print the columns of X DataFrame
print(X_df)
```


```python
# dataset
dataset = pd.read_csv('cleaned data after select features.csv')
# distributing the dataset into two components X and Y
X = dataset.iloc[:, 0:17000].values
y = dataset.iloc[:, 9].values
print(X.shape)
#performing preprocessing part
sc = StandardScaler()
X = sc.fit_transform(X)
le = LabelEncoder()
y = le.fit_transform(y)
# Applying PCA function on training
# and testing set of X component
pca = PCA(n_components = 5)
X = pca.fit_transform(X)
print(X.shape)
# Convert numpy arrays to pandas DataFrame
X_df = pd.DataFrame(X, columns=['PC1', 'PC2','PC3','PC4','PC5'])
# Print the columns of X DataFrame
print(X_df)
```


```python
# Print the variance explained by each principal component
explained_variance = pca.explained_variance_ratio_
print('explained_variance: ' + str(explained_variance))
# Plot the transformed data first 2 columns with the highest variance
import matplotlib.pyplot as plt
plt.scatter(X[:, 0], X[:, 1], c=y)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.show()
# saving the dataframe into a csv file
X_df.to_csv('features_after_reduction.csv')
```

[10]

```
(17023, 10)
(17023, 5)
```



PCA—>Principal component analysis, or PCA, is a dimensionality reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data points much easier and faster for machine learning algorithms without extraneous variables to process.

code:

https://github.com/Ms850446/ECG_Appnea_Feature_Extraction/tree/main