ARTIFICIAL INTELLIGENCE AND DATA ENGINEERING

Project for the Course of

DATA MINING AND MACHINE LEARNING

Emotions Classification through Wearable Sensor Data and Analysis of the Feature Importance

Students:

Martina Burgisi **Daniele Giaquinta**

ACADEMIC YEAR 2022/2023

Index

Abstract

Automated Emotion Evaluation is a key theme in various fields that use human emotional reactions as tools; in marketing to create special advertisements based on the emotional state of the potential customer, in education to improve learning processes, in technical equipment to ensure better human-machine interaction and many others. Processing and understanding human emotions falls within the broad area of what is known as "Affective Computing".

In this project, the K-EmoCon dataset will be used [1], containing information on human emotions that arise during social interactions; in particular, this multimodal dataset contains samples of the most common mobile bio-signal sensors collected during conversations of 32 subjects who participated in the experiment. At the end of these exchanges, the subjects described the sensations they had experienced according to the emotional model defined by James A. Russell [2], which will be discussed later in the document. Techniques of machine learning will be used to look for correlations between the data collected by the sensors and the sensations experienced by the participants. The classifiers will make predictions on the values of the fields defined by the emotional model. An accuracy score will be assigned for these predictions and the same procedure will be repeated after applying a dimensionality reduction technique of the data collected by the sensors, the PCA, and it will be noted that the accuracy scores will be slightly affected but this is a toll to pay in order to improve explainability. These classifiers are however defined as "black box", it is not simple to understand the types of reasoning that they have followed in the formation of their patterns and in the subsequent predictions. The second part of the document will therefore try to classify the importance of the data and how they are used to make predictions (both in the integrated format and in the consolidated format); two different techniques of feature importance analysis will be adopted, one to understand the general importance of the features in the model as a whole and another one to explore what was considered the most important during the prediction for any given row of data.

Introduction

Affective computing interfaces with and arises from emotions. It is a multidisciplinary field that is constantly growing and explores how technology can infer understanding of emotions, how interactions between humans and devices of all kinds can be influenced by emotional states, how systems can be designed to use this data and improve their abilities, and how affective and sensory strategies can transform the interaction between man and computer. Disciplines that embrace it are engineering, psychology, education, cognitive science, sociology and many more.

However, it is inevitable to take several steps back; to achieve these goals, it is necessary that computers or machines in general have emotional intelligence, that is, they are able to recognize and understand human feelings and emotions in order to behave accordingly. Once again, in order to even recognize these emotional states, machines must learn to do so; teaching this to a machine is a difficult task because even our scientific knowledge about these topics is limited, uncertain, constantly evolving and overturned. With these problems to keep in mind, the usefulness of a machine learning-based approach emerges. However, to follow this path, a very large dataset is needed, something that in the field of neuroscience is always very rare as each study never manages to be carried out on a sufficiently large number of subjects or with a collection of data significant enough to ever give us definitive answers.

That's why the dataset on which the work will be done is so important; not only is it one of the first publicly available to boast a large volume of data, but it is also multi-perspective and therefore contains annotations on the sensations experienced during interactions between two people both by the person concerned and by the interlocutor and by external observers. More specifically, the experiment that led to the creation of K-EmoCon consists of 16 conversation sessions (for a total of 32 participants) each lasting 10 minutes with topics of social dynamics. Each participant wore four devices during their sessions: the "Empatica E4" bracelet, the "Polar H7" heart rate sensor, the "NeuroSky" headset for brain waves and finally a camera that filmed the interlocutor. In addition to audio and video information, EEG, photoplethysmography, heart rate, blood volume pulse, inter-beat interval, body temperature, electrodermal activity, three-axis accelerometer, attention rate and relaxation level data were collected for each subject. All this is accompanied by

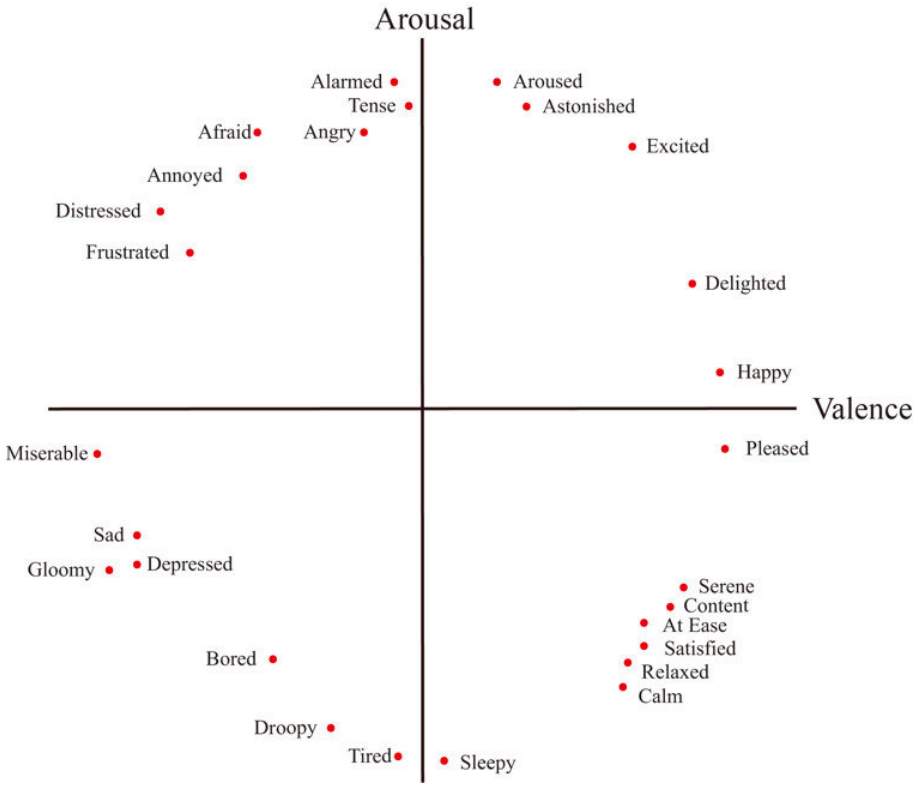

Figure 2.1: James A. Russell's circumplex model of affect

personal statements, the partner's and external statements on the emotions felt by each person and the interlocutor during the interaction.

For this last task, the previously mentioned model by James A. Russell was adopted, which distributes human emotions on two axes; on the Y-axis is measured arousal, that is the degree of stimulation experienced, on the X-axis is measured the valence, that is the degree of pleasantness of the interaction. Combinations of different values in these two domains identify different emotional states Figure 2.1.

At the end of each session, the sensations experienced on the valence-arousal scale were described by each subject, by their interlocutor and by external observers, assigning these two variables with integer values in a range from 1 to 5. 6 variables were thus produced and also columns of the dataset that also represent the possible classification targets Figure 2.2.

The experiment of this project will focus in particular on the target attributes *self valence* and *self arousal* (the pleasantness and stimulation of the conversation perceived by the subject). It is reasonable to expect that the subject of the interaction is the one who knows best the sensations experienced; analyzing a sensation evaluated by the subject therefore allows to limit a possible absence of pattern between this and the feature due to the incorrect interpretation of the subject by the interlocutor or external observers.

As already mentioned, the subjects wore 3 multi-sensor devices and each sensor performed multi-modal samples with a certain frequency; these were: the "Empatica

E4" bracelet, the heart rate device "Polar H7" and the "NeuroSky" headset for brain waves. These devices produced 18 variables and features for classification Figure 2.3:

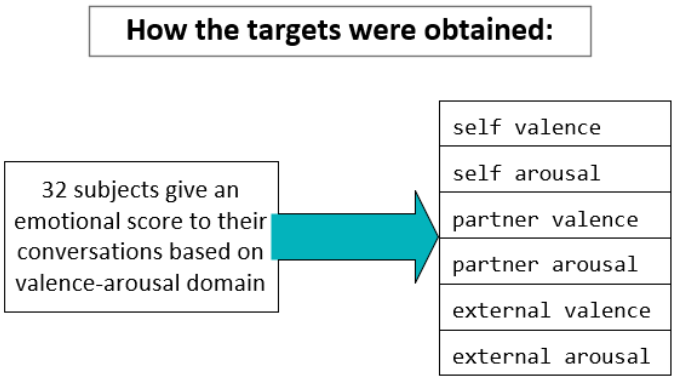

Figure 2.2: Overview of all possible target attributes of the classification

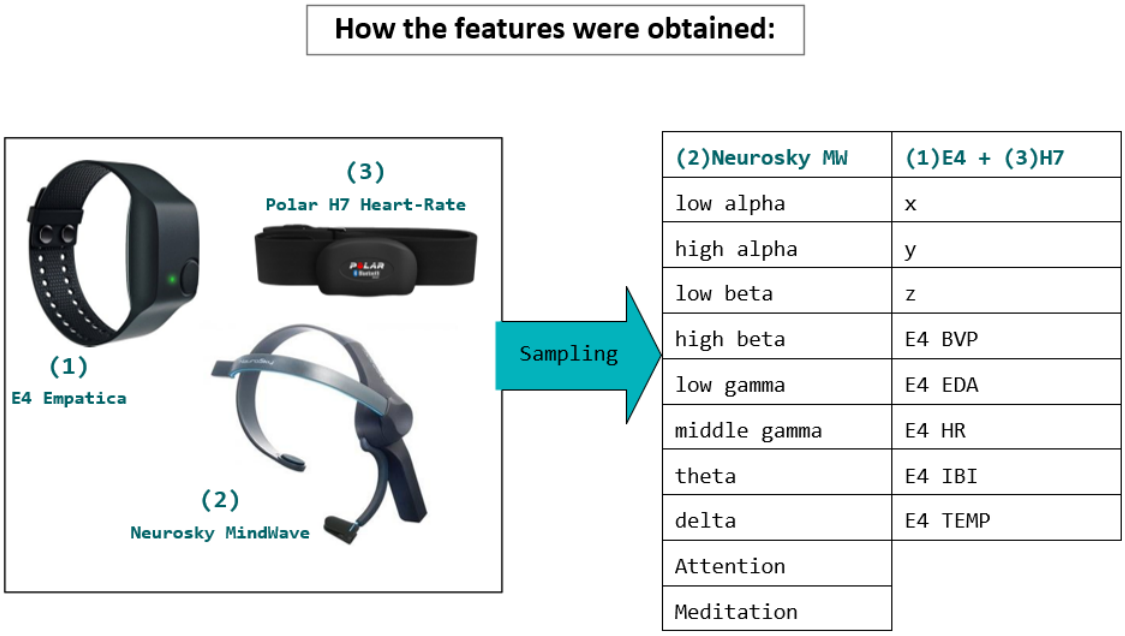

Figure 2.3: Overview of the feature attributes sampled by the sensors

For reasons that will be discussed in the preprocessing chapter, **the data relative to the E4 IBI sensor has been cut from our model**. Now follows a brief description of the kind of data gathered by each sensor:

- x, y, z are the measurement axes of the accelerometer
- $E4\ BVP$, $E4\ HR$, $E4\ IBI$ are respectively blood volume pulse, heart rate, inter-beat interval, all obtained through photoplethysmography
- $E4\ TEMP$ is the body temperature

- *E4 EDA* is the electrodermal activity
- *low alpha*, *high alpha*, *low beta*, *high beta*, *low gamma*, *middle gamma*, *theta*, *delta* are the 8 frequency bands on which brain activity is measured
- *Attention*, *Meditation* are also frequencies measured by the headset and indicate the rate of focus and relaxation of the subject

Data Overview

The preprocessing phase for the data was quite laborious due to the way they were divided; the dataset in fact contains the following 5 folders. Figure 3.1:

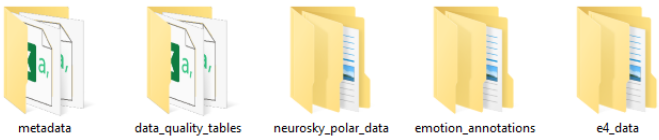

Figure 3.1: Folders composing the dataset

Inside the *metadata* folder are contained 2 CSV files:

- **data_availability.csv** - indicates for each subject the availability of a given sensor data and a given emotional annotation
- **subjects.csv** - indicates for each subject the timestamp at which they entered the room, the timestamp at which the session began, and the timestamp at which it ended

Inside the *data_quality_tables* folder are contained 7 CSV files:

- **e4_durations.csv** - contains the duration of each fle in seconds, where duration = (last timestamp - first timestamp) / 1000
- **neuro_polar_durations.csv** - same as above
- **e4_zeros.csv** - contains the number of zero values in each fle.
- **neuro_polar_zeros.csv** - same as above
- **e4_outliers.csv** - contains the number of outliers in each fle. Chauvenet's criterion was used for outlier detection
- **e4_completeness.csv** - contains the completeness of each fle as a ratio in the range of [0.0, 1.0]. 1.0 indicates a fle without any missing value
- **neuro_polar_completeness.csv** - same as above

Inside the *emotion_annotations* folder are contained 4 subfolders:

- **self_annotations** - contains the emotional annotations each subject gave about themselves
- **partner_annotations** - contains the emotional annotations each subject gave about their partner
- **external_annotations** - contains the emotional annotations given by external observers about each subject
- **aggregated_external_annotations** - same as above in aggregated form

We will only analyze self annotations with the assumption that the feedback left on one's own emotions is more reliable than that left by external observers on the same. **The folder self_annotations contains 32 CSV files, one for each subject, containing the scores in the values-arousal domain** Figure 3.2:

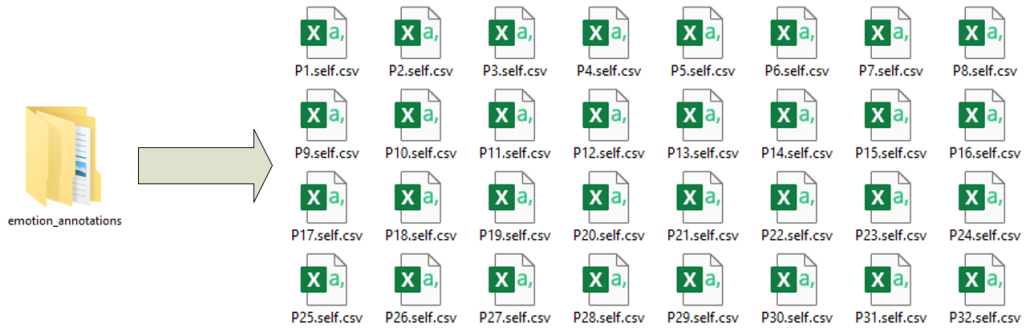

Figure 3.2: Content of the self_annotations subfolder

Inside the folders e4_data and neurosky_polar_data there are 32 subfolders, one for each subject Figure 3.3:

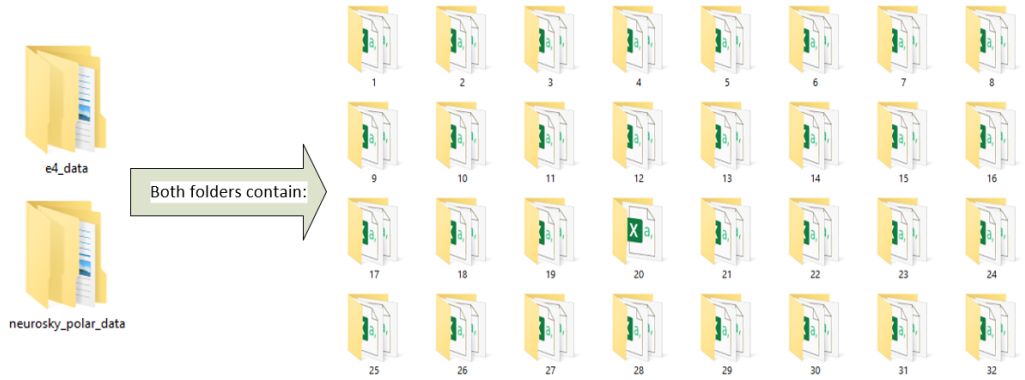

Figure 3.3: Content of the e4_data and neurosky_polar_data subfolders

In each one of the numbered subfolders are contained the samplings of the sensors Figure 3.4:

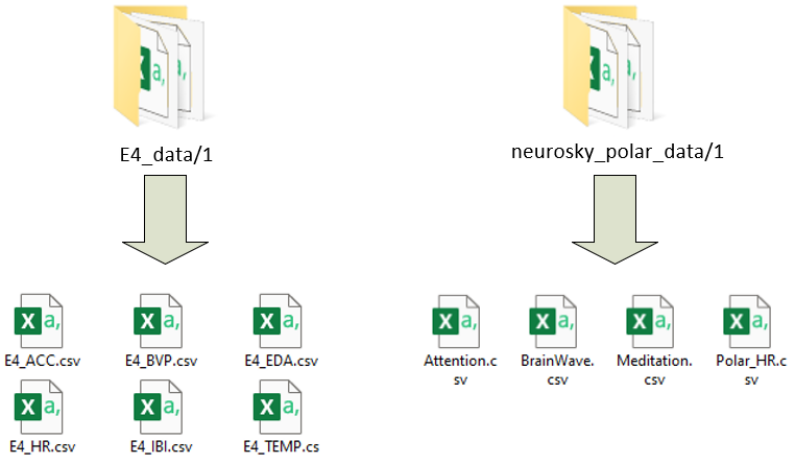

Figure 3.4: Content of each numbered subfolder

Preprocessing

Now the steps followed for the data preprocessing that led to obtaining a single aggregated table will be briefly explained.

Preliminary Considerations

It is important to note that we could not use the data relative to every subject due to the incompleteness of it. The subjects identified by pids 1, 2, 3, 6, 7, 20, 26, 29, 30, 31, 32 were excluded for this reason; keeping them would not allow a uniform ingestion of the data due to missing labels and missing samplings. We also didn't include all the data from the sensor named E4_IBI for the same reason; its samplings didn't usually match the temporal ranges of the sessions and keeping it would have costed us approximately 40% of the dataset rows due to the impossibility of keeping empty cells.

Timestamp Conversion and Horizontal Merge of the Sensor Data

First of all, it is necessary to convert the column that provides a temporal information in all tables. This is in fact saved in timestamp format but the emotional annotations are collected at 5-second intervals; for aggregation, a conversion to datetime format is therefore necessary. **The first part of the preprocessing code reads csv files and converts timestamp columns to datetime objects, then writes the modified data back to the csv files and finally all the CSVs in each numbered folder are merged horizontally using the converted timestamp as key**; more specifically:

- It defines a function 'convert_timestamp_to_datetime' that takes a timestamp in string format, removes the decimal point and any trailing digits, converts it to seconds and returns the datetime object.
- It then iterates through subfolders in two different data folders, 'e4_data' and 'neurosky_polar_data'. For each file found in these subfolders, it opens the file in read mode, reads the rows and stores them in a list.
- The code then opens the file in write mode and writes the column headers back to the file.

- Then, it iterates through the rest of the rows and converts the timestamp in the first column of each row to a datetime object using the previously defined function. It then replaces the timestamp with the datetime object in the row and writes the modified row back to the file.
- The code then repeats the same process for 'subjects.csv' in the metadata folder; this is the timestamp relative to the beginning of the sessions.
- Lastly, the data in each numbered foder is merged horizontally using the converted timestamp as key. Not all sensors have the same sampling frequency; empty cells were fitted with the last valid value.

Processing the Chunks

The CSV files obtained as explained retain useless information like the serial numbers of the sensors and useless rows like the ones out of the conversation sessions temporal range. The next step was to fix these issues.

The code then begins by cutting unnecessary columns and renaming the remaining columns to more meaningful names for the data in the 'e4_data' folder. It iterates over subfolders in the specified path, reads the csv file 'e4_merged.csv' into a DataFrame, selects only certain columns and renames them, then writes the modified dataframe to a new csv file 'e4_processed.csv'.

Then, it creates a dictionary that stores the start and end timestamps for each subfolder, and uses this data to filter rows by only including 10 minutes of data starting from the beginning of the conversation for each subfolder in the 'e4_data' folder. It iterates through each subfolder, reads the 'e4_processed.csv' file into a DataFrame, filters the rows to only include those between the start and end timestamps for that subfolder, then writes the modified dataframe to a new csv file 'e4_filtered.csv'.

The same operations are executed for the neurosky_polar_data folder.

Vertical Merge of the Tables

All these preprocessed data are still split into each numbered subfolder for both the e4_data folder and the neurosky_polar_data folder. **The next step is to vertically merge the data for every subject.**

The code vertically merges the processed E4 data from every participant and saves it to a new CSV file. It does this by initializing an empty list, iterating over each numbered subfolder in the E4 data directory, reading the 'e4_filtered.csv' file in each subfolder, appending the dataframe to the list, concatenating all the dataframes into a single dataframe, and then saving it to the new CSV file. The same tasks are performed in the neurosky_polar_data folder.

Horizontal Merge of the Device Data

At this point we possess one CSV file with all the E4 sensor data for every subject and one CSV file with all the neuroky polar sensor data. **These two files are to be merged horizontally**. The code does this by reading the 'e4_merged.csv' and 'neurosky_merged.csv' files into pandas dataframes, truncating the timestamp column in both dataframes to the nearest second, performing an outer merge, filling missing values with the previous row's data, and then saving the merged dataframe to a new CSV file.

Final Horizontal Merge with the Emotion Annotations

For the last task of the preprocessing we need to **horizontally merge the whole samplings data with the emotional feedbacks left by the subjects.** The code achieves this by converting the 5 seconds interval of the emotional annotations in datetime timestamps to facilitate the merge with the sensor data. It does this by defining a function 'convert_seconds_to_datetime' that takes in seconds and an index value, creating a base datetime based on the value of the index, adding the seconds to the base datetime, and returning the resulting datetime.

Classification

The next step of our project consisted of course in classifying the values of valence and arousal of which the possible classes are {1, 2, 3, 4, 5}. We used different classifiers in order to pick the best model and use it for our purpose. **Our case-study represents a 2-output 5-class scenario**, so the accuracy scores that will follow must be interpreted by also taking this into account.

5.1 Classifiers Comparison

We built 4 models using the following classifiers:

- **K Nearest Neighbors:** Accuracy Score = 0.497
- **Naive Bayes:** Accuracy Score = 0.447
- **Decision Tree:** Accuracy Score = 0.588
- **Random Forest:** Accuracy Score = 0.931

The chosen model is the one using Random Forest. In order to further investigate the performance of the model we plotted the confusion matrix for both target attributes and we printed the classification report Figure 5.1 Figure 5.2.

Table 5.1: Arousal classification report

Label	Precision	Recall	F1-score	Support
1	0.99	0.99	0.99	428
2	0.99	0.93	0.96	2334
3	0.91	0.99	0.95	3978
4	0.98	0.90	0.94	2206
5	0.99	0.99	0.99	601

- **accuracy:** 0.95 - **macro avg:** 0.97 - **weighted avg:** 0.96

Table 5.2: Valence classification report

Label	Precision	Recall	F1-score	Support
1	1.00	0.98	0.99	230
2	0.95	0.80	0.87	1423
3	0.92	0.92	0.92	4922
4	0.81	0.89	0.85	2697
5	0.99	0.96	0.98	275

- **accuracy:** 0.89 - **macro avg:** 0.93 - **weighted avg:** 0.90

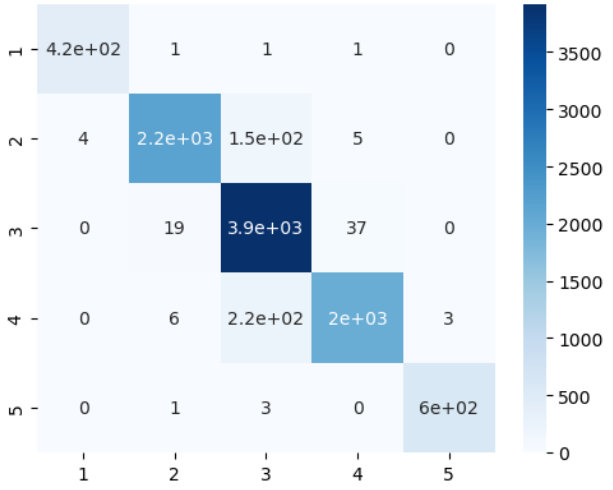

Figure 5.1: Confusion Matrix for Arousal

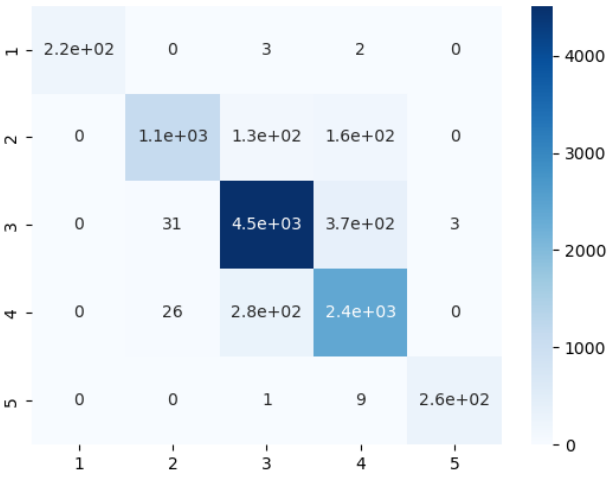

Figure 5.2: Confusion Matrix for Valence

The scores are quite high in both reports and the classification can be considered more than satisfactory. In the confusion matrices the predicted classes are displayed on the x-axis, and the actual classes are displayed on the y-axis so we can notice how the error in the prediction phase is distributed.

5.2 Principal Component Analysis

The next phase of our project involves the use of Principal Component Analysis to reduce the number of variables, that initially are 17. However, the criterion with which we want to perform this reduction is that of interpretability; our idea is to merge the columns related to sensors that perform measurements on the same type of biosignals; more specifically Figure 5.3:

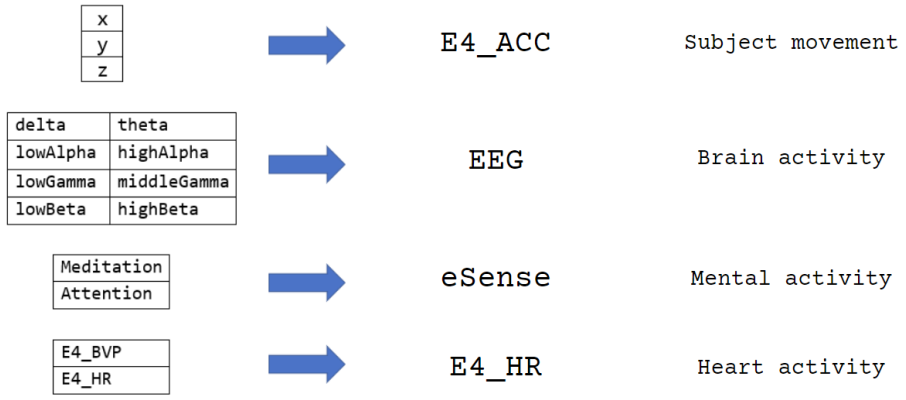

Figure 5.3: Overview of the columns that have been aggregated together

On the left the starting columns, in the middle the name of the new columns and **on the right an interpretation of the meaning of the aggregated columns**. Note that **two of the columns (E4_TEMP and E4_EDA) were used in the reduced model without being aggregated with any other column** due to a lack of semantic similarity with other sensors.

Correlation Matrix

Our choice for aggregation through PCA only adopts a criterion of interpretability and meaning of the attributes, but it could prove to be a wrong choice from a more strictly mathematical point of view. Remember that when PCA takes n parameters in input (or rather columns) and reduces them to a single one, it only changes the point of view from which these are "observed" all together, adopting the one that preserves the maximum covariances possible. However, if the variables that are aggregated are highly uncorrelated and develop in completely different directions (the worst case is perpendicular), PCA makes us lose information.

A way to try to validate or refute our choice is to visualize the correlation matrix of the data, significant since we only have numerical and continuous columns. If it

were to result that in the columns chosen by us for union through PCA there was a not indifferent correlation, then our choice would be revealed as correct. This is in fact what happens; the correlation matrix reveals precisely the correlations between the columns that sample semantically similar biosignals Figure 5.4:

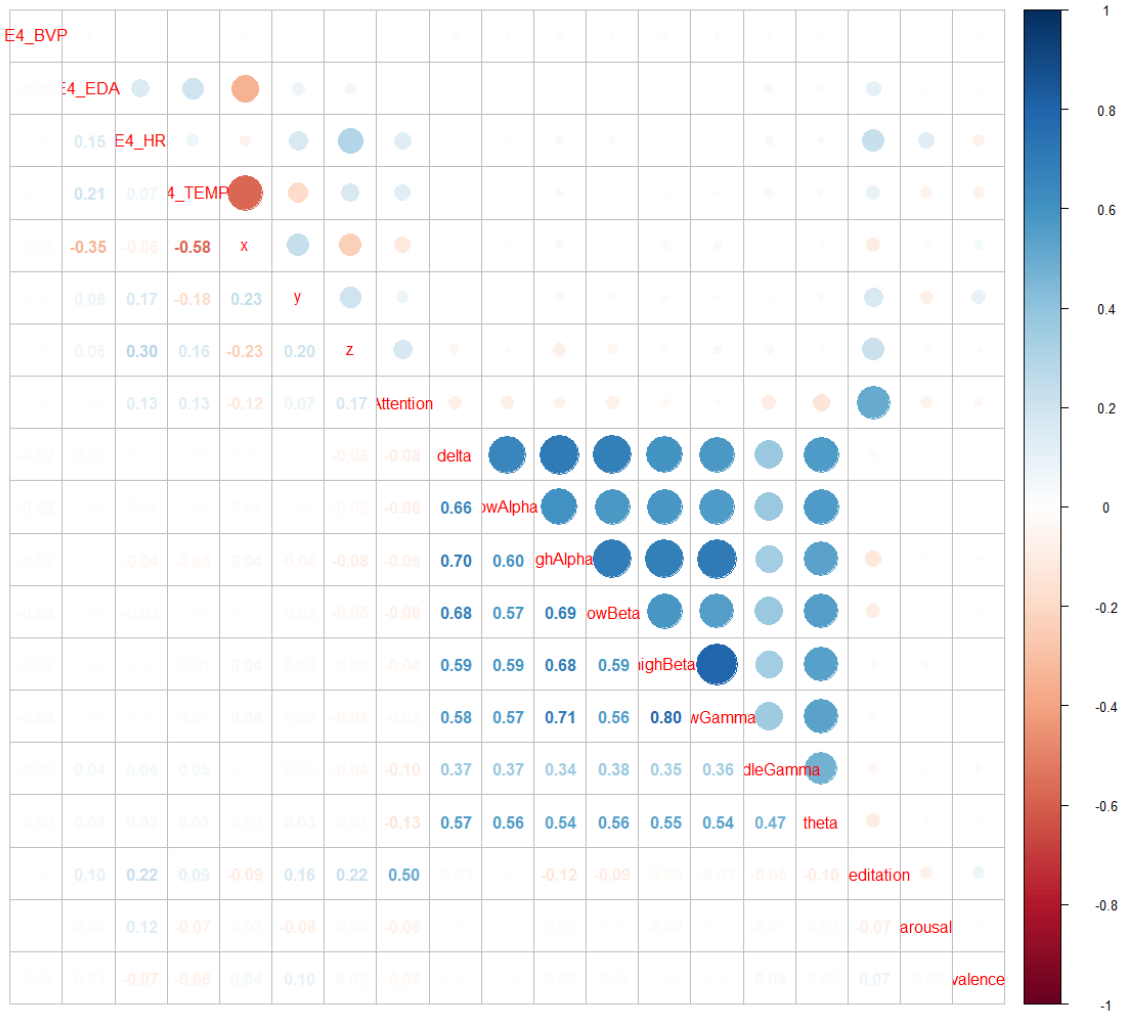

Figure 5.4: Correlation Matrix of the Data

5.2.1 Classification after PCA

For completeness, we created four models after PCA, one for each of the previously mentioned classifiers, obviously without expecting the situation to change:

- **K Nearest Neighbors:** Accuracy Score = 0.596
- **Naive Bayes:** Accuracy Score = 0.492
- **Decision Tree:** Accuracy Score = 0.612
- **Random Forest:** Accuracy Score = 0.903

We can notice that every classifier other than Random Forest benefitted from the Principal Component Analysis, of course the Random Forest still has the upper hand by a very significant margin. It is possibile that this improvements comes from the scaling of the variables which is preparatory for the application of the PCA technique.

The main reason why we did not scale the columns in the full model is interpretability: scaling would not allow the user to adjust the samplings of the sensors precisely and understanding the impact of the biosignals on emotions would be harder.

Random Forest is less affected by the scaling of the columns so the score after PCA decreased due to the probable loss of a small proportion of information. **Overall it is still the chosen classifier by far.**

Feature Importance

The last phase of our project involves explaining the reasoning carried out by the chosen model, both in the case without PCA and in the case with PCA. We have decided to analyze two cases: the general importance of the features in the entire model and the importance of the features in predicting a certain input tuple.

6.1 General Feature Importance

In order to visualize the feature importance of the whole model (for both scenarios) we used the `feature_importances_` utility from scikit-learn.

- The feature importances are calculated by the model, which assigns an importance score to each feature based on how much the feature contributed to the predictions.
- The code first gets the feature importances from the Random Forest classifier model using the `feature_importances_` attribute.
- The code then sorts the feature importances in descending order using the numpy function `np.argsort(importances)[::-1]`. This gives an array of indices that correspond to the order of the features by importance.
- The code creates a list of feature names that correspond to the sorted indices using a list comprehension. This rearranges the feature names so they match the sorted feature importances.
- The code creates a bar plot of the feature importances using the `plt.bar()` function. The x-axis of the plot is the range of the number of features and the y-axis is the feature importances sorted in descending order.

How the Scores are Calculated

The code uses the **feature_importances_** attribute of the model, which returns an array of importance scores, one for each feature in the dataset. These importance scores are calculated using the MDI (Mean Decrease Impurity) method.

MDI measures the decrease in impurity of a decision tree when a feature is used to split the data. The more a feature reduces impurity, the more important it is deemed to be. In Random Forest, the feature importance is calculated as the average feature importance over all decision trees.

In the case of Random Forest the Gini measure for impurity is used; Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. The Gini impurity is calculated as the probability of a random element being classified incorrectly if it was randomly labeled according to the class distribution in the subset. A Gini impurity of 0 corresponds to a pure subset, while a Gini impurity of 1 corresponds to a subset where the elements are split evenly across all classes.

We obtained the following plots for the feature importance for the two models Figure 6.1 Figure 6.2:

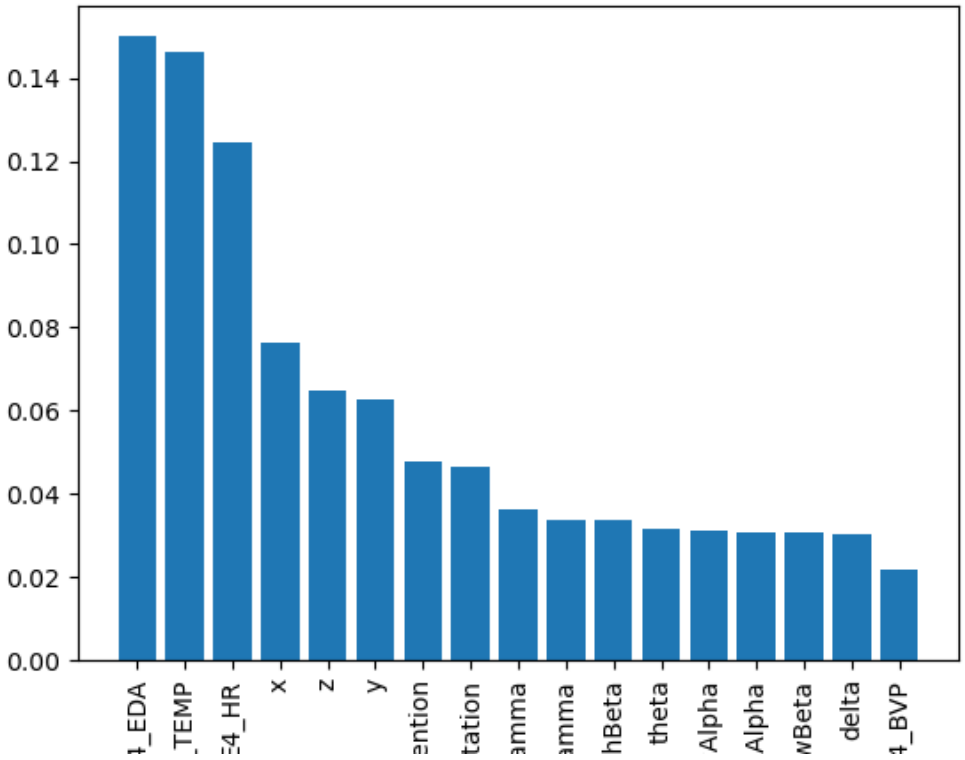

Figure 6.1: General Feature Importance of the Model

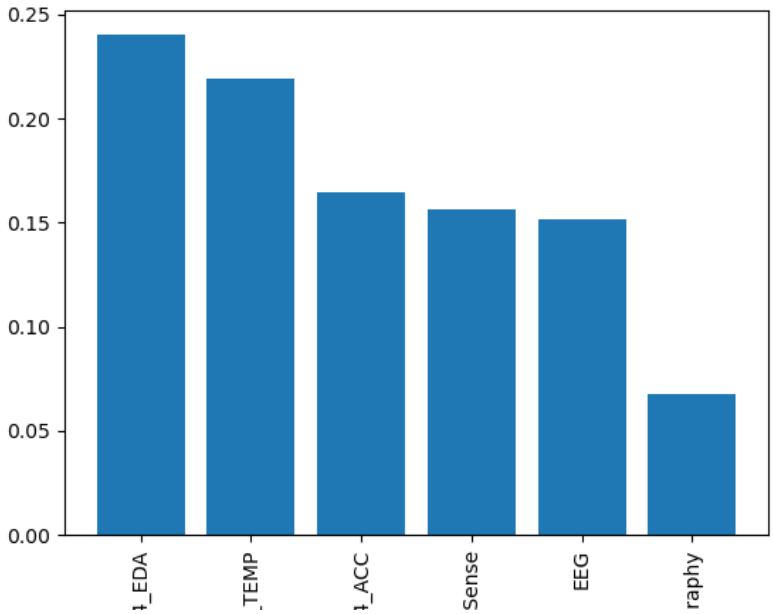

Figure 6.2: General Feature Importance of the Model after PCA

6.2 Tuple Feature Importance

Before discussing how the feature importance for a certain input tuple was analyzed, it is important to remember that a classification model built with Random Forest is referred to as a "black box" model; apart from the general feature importance computed as described above, it is impossible for us to get exact feature importance for the prediction of a tuple since it is not possible to scrutinize how the running model thinks.

For this reason, **for feature importance analysis for an input tuple we used SHAP** (SHapley Additive Explanations), which has the characteristic of being model agnostic, that is, it estimates feature importance by mathematical techniques without the need to know the model.

How the Scores are Calculated

SHAP works by computing the contribution of each feature to the prediction for a particular input. For a classification model, it calculates the contribution of each feature for each class for the specific input.

It does this by approximating the model locally around the input point and then computing the expected value of the feature's contribution based on all possible coalitions of features. These contributions are then used to compute the SHAP values, which are a measure of how much each feature contributes to the prediction.

By looking at the SHAP values, you can understand how the model is making its predictions and identify which features are the most important for a given input.

You can also identify any feature interactions that are important for the prediction, and detect any bias in the model.

Now are presented two examples of SHAP explainations Figure 6.3 Figure 6.4:

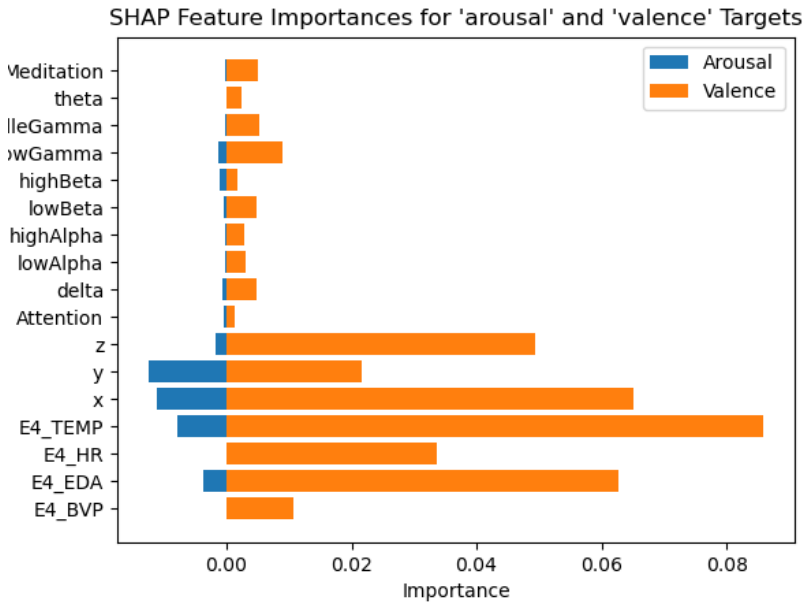

Figure 6.3: Example of SHAP feature importance of a tuple

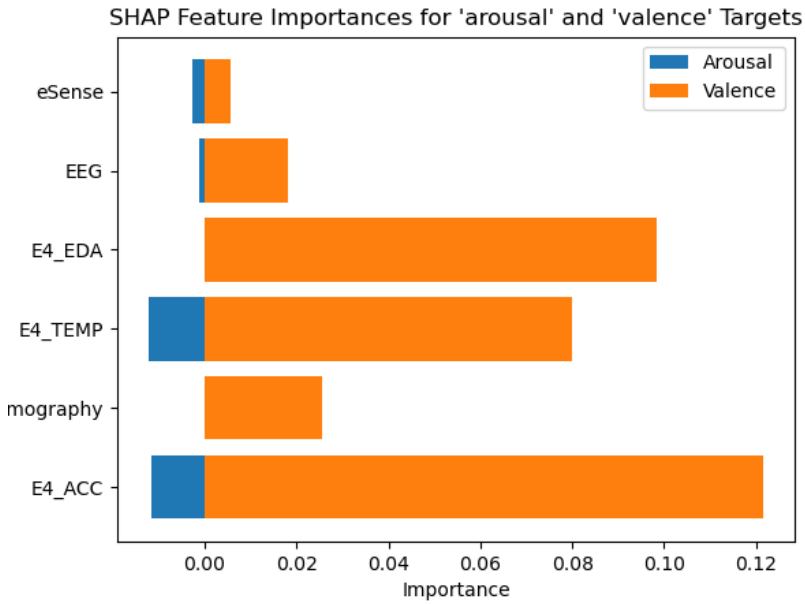

Figure 6.4: Example of SHAP feature importance of a PCA tuple

GUI

The application created is a web app that has an interface consisting of an area containing a series of sliders; three buttons (Instruction Manual, PCA Components Mode, Submit); and an output area. The latter part provides two horizontal sliders, representing Valence and Arousal values, and an "Emotion" field, used to show the corresponding emotion through a word. There is also a "Feature Importance" button.

Features are represented by the vertical sliders, which can be edited to set a specific value. It is possible to set the value of each slider by scrolling the slider with the mouse, or by editing its numerical value visible below it: after selecting it, it is necessary to enter the desired value and then press the enter key. If the value entered is within the allowed range, the slider will be updated; otherwise, an error message will be shown in which the domain of slider use is made explicit.

The targets are represented by two horizontal sliders (Valence and Arousal) whose values range from 1 to 5.

The "Instruction Manual" button displays a block containing a brief description of the dataset origin and instructions on how to use the application. The "PCA Components Mode" button allows you to switch to the "PCA Components" mode and then back to the original mode. The PCA Components mode allows you to use the model obtained with PCA, and consequently to use the sliders related to the components obtained with the aforementioned analysis. As previously mentioned, the components obtained are E4_ACC, E4_EDA, E4_Fotoplethysmography, E4_TEMP, EEG, and eSense with the goal of obtaining one feature per sensor. The management of these sliders is identical to the sliders related to the full feature set.

The "Submit" button allows the sliders' set values to be used to predict Valence and Arousal using the model. In addition, the submit operation also allows you to get the corresponding emotion in the right pane of the Valence and Arousal sliders, and the graphs related to the Feature Importance Analysis, which you can view by clicking on the "Feature Importance" button. The "Feature Importance" button opens a block where the user can view the General Feature Importance Plot and the Row Feature Importance Plot related to the last submit operation and the currently used mode.

7.1 Implementation

The application was built using HTML, CSS and Javascript languages. In particular, the vertical sliders are dynamically loaded using Javascript, thus managing the transition from one mode to another.

There are three folders: data, static and templates. The data folder contains the data, the composition of which was previously mentioned in the Data Overview chapter; the static folder contains the img, css and js subfolders, which in turn contain the related files necessary for the application to function. Specifically:

- **style.css** - contains the CSS code that allows you to apply style properties to the html document;
- **buttonsProperties.js** - contains the Javascript code needed to apply and manage the functionality of each button. This script also contains the code needed to make Ajax calls and to handle the server response. Specifically: when the "Submit" button is pressed, an Ajax request is made to obtain the predicted Valence and Arousal values. Once obtained, a constant matrix is used into which the emotions corresponding to these Valence and Arousal values have been mapped using the James Russel model. An additional Ajax request is made at the time the "Feature Importance" button is pressed, to obtain the corresponding "Row Feature Importance Plot" and display it along with the "General Feature Important Plot" in the block created dynamically with Javascript.
- **slidersProperties.js** - contains the Javascript code needed to apply and manage the functionality of each slider.

The templates folder contains just one file:

- **index.html** - it contains the static part of the html code, through which the interface is built.

7.1.1 Flask

On server side, abbiamo usato Flask. Flask is a web framework. This means flask provides tools, libraries and technologies that allow to build a web application. Flask is part of the categories of the micro-framework. Micro-frameworks are normally framework with little to no dependencies to external libraries.

Flask allowed us to run the Python code needed to build the model and obtain the predicted Valence and Arousal values in both the PCA and general case. Specifically:

- **app.py** - file to run to start the server.
- **views.py** - file whose main content is a function to handle Ajax calls.

- **classify.py** - file that contains the code needed to build the model as previously mentioned. It contains additional functions useful to get information about the components (features) and to get the row feature importance plot and PCA row feature importance plot.

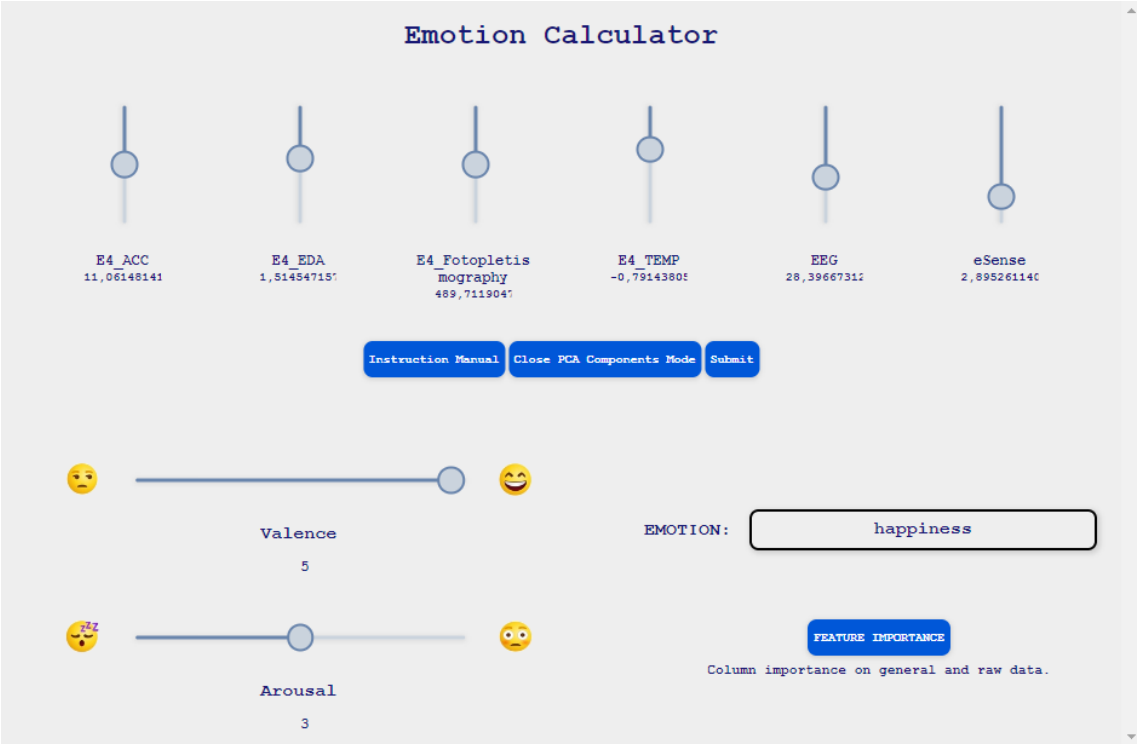

Figure 7.1: Overview of the graphical user interface

Bibliography

[1] C. Y. Park, N. Cha, S. Kang, A. Kim, A. H. Khandoker, L. Hadjileontiadis, A. Oh, Y. Jeong, and U. Lee. K-EmoCon, a multimodal sensor dataset for continuous emotion recognition in naturalistic conversations. *Sci Data*, 7(1):293, 09 2020.

[2] J. Posner, J. A. Russell, and B. S. Peterson. The circumplex model of affect: an integrative approach to affective neuroscience, cognitive development, and psychopathology. *Dev Psychopathol*, 17(3):715–734, 2005.