# Machine Learning project for psychological state prediction

Computer Engineering and Robotics – Academic Year 2024-2025

Salvatori Gianluca

# 0. Contents

# 0. List of Figures

# 1. EDA

The dataset comprises information gathered from biosensors and the surrounding environment of individuals, mostly students, during specific activities. The purpose purpose of this data collection is identify the psychological states and level of stress of the subjects.
A comprehensive description of the dataset's features can be found on it's Kaggle page:[1]

For the part of exploratory data analysis, the dataset was examined for null and duplicated values revealing that none were.

```python
print(df.isnull().sum())
print(df.nunique().sum())
```

Furthermore, for the purpose of this project, the features "ID" and "Date" have been deemed irrelevant, and were therefore eliminated.

```python
df = df.drop(['ID'], axis=1)
df = df.drop(['Time'], axis=1)
```

The features "EEG Power Bands" and "Blood Pressure (mmHg)" contain valuable information that must be preserved. However these features are stored as Object variables.
For this project, I opted to extract the values in the strings and to save them as new, distinct features. The three values of EEG Power Bands were extracted and stored in the features "EEG_Delta," "EEG_Alpha," and "EEG_Beta.". Similarly, the two values of "Blood Pressure" were stored in the features "Pressure_Systolic" and "Pressure_Diastolic."

To accomplish this two functions were created *Extract_values* and *split_values*. The first one returns the EEG Power Bands values and use the method *ast.literal_eval()*, that gets in input a literal, and return the list of elements that compose it.

The new columns of the dataset are created and then populated with the corresponding values by applying the defined functions. Finally the original features were dropped from the dataset.

```python
def extract_values(row):
    return ast.literal_eval(row)

def split_values(row):
    a, b = row.split("/")
    return int(a), int(b)

df['Bands'] = df['EEG Power Bands'].apply(extract_values)
df['EEG_Delta'] = df['Bands'].apply(lambda x: x[0])
df['EEG_Alpha'] = df['Bands'].apply(lambda x: x[1])
df['EEG_Beta'] = df['Bands'].apply(lambda x: x[2])
df['Pressure_Systolic'] = df['Blood Pressure (mmHg)'].apply(lambda x: split_values(x)[0])
df['Pressure_Diastolic'] = df['Blood Pressure (mmHg)'].apply(lambda x: split_values(x)[1])

df.drop(['Blood Pressure (mmHg)','Bands','EEG Power Bands'],axis=1,inplace=True)
```

Lastly, the target feature's data type were converted from string to integer, and encoded using zero-based indexing.

In the project I developed two different scripts: one for the prediction of the emotional state, and the other for the prediction of the cognitive load. This approach was designed to compare the different performances of the two variants.
The primary difference between the two scripts lies in the encoding of the target variable and the creation of the Design Matrix **X**.

For the psychological state prediction:

```
df.replace({'Psychological State': 'Stressed'}, 0, inplace=True)
df.replace({'Psychological State': 'Relaxed'}, 1, inplace=True)
df.replace({'Psychological State': 'Focused'}, 2, inplace=True)
df.replace({'Psychological State': 'Anxious'}, 3, inplace=True)
pd.set_option('future.no_silent_downcasting', True)

t = df['Psychological State'].values
X = df.drop(['Psychological State'],axis=1)
```

For the cognitive load prediction:

```
 df.replace({'Cognitive Load': 'Low'}, 0, inplace=True)
df.replace({'Cognitive Load': 'Moderate'}, 1, inplace=True)
df.replace({'Cognitive Load': 'High'}, 2, inplace=True)
pd.set_option('future.no_silent_downcasting', True)

t = df['Cognitive Load'].values
X = df.drop(['Cognitive Load'],axis=1)
```

The final step involved checking the correlation between the features using a correlation matrix, which revealed no significant correlations.1.1

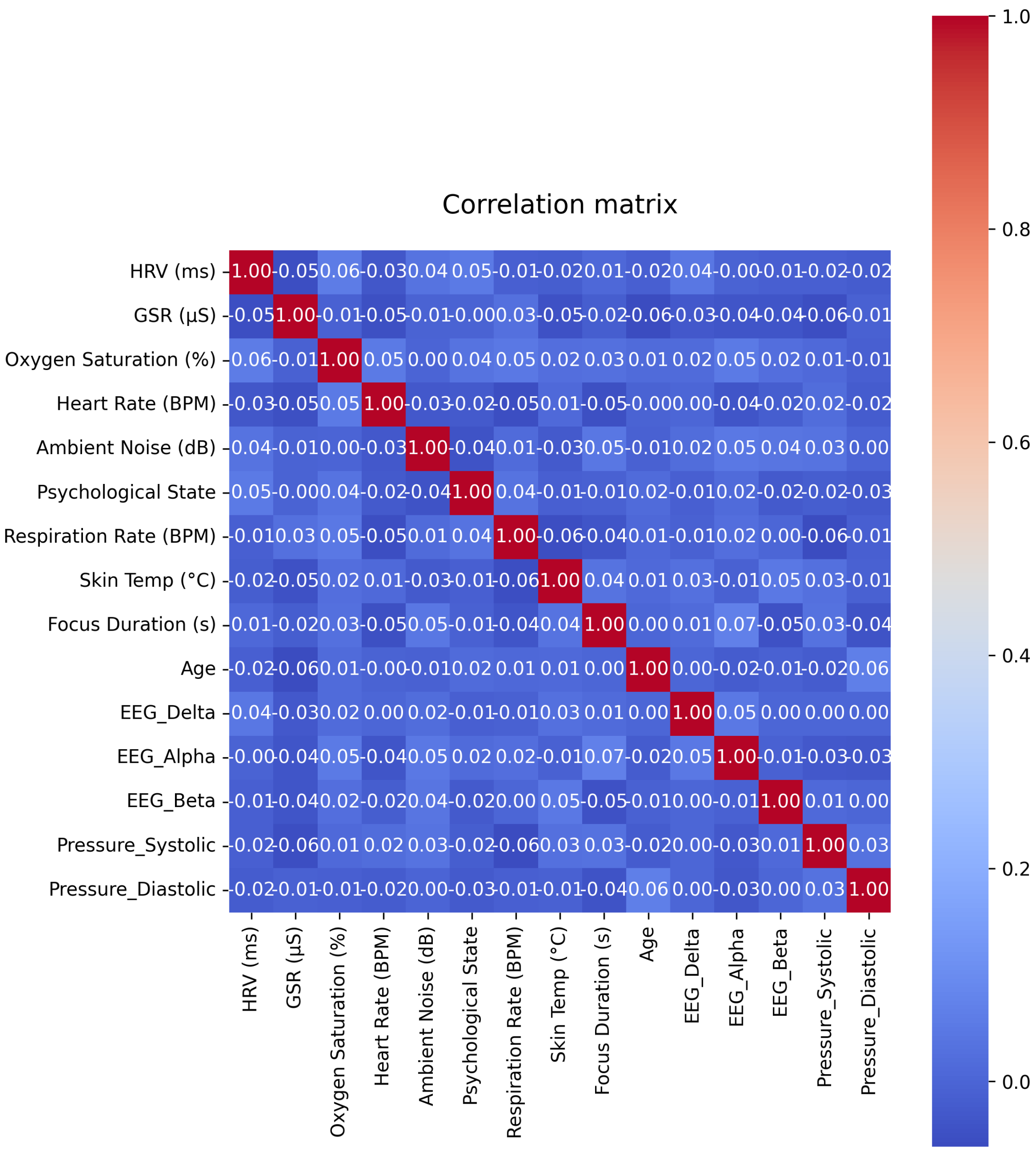



Figure 1.1: Correlation Matrix

# 2. Model selection and Model assestment

For this project three different classifier were selected: logistic regression, SVM and a Neural Network.
Each classifier was tuned by testing several configuration of its parameters.

Initially, the entire datasetwas divided into Train, Development and Test sets. Subsequently the data was normalized with a standard scaler

```
from sklearn.model_selection import train_test_split
X_train, X_test, t_train, t_test = train_test_split(X, t, test_size=0.2,
                                                    random_state=42)
X_train, X_dev, t_train, t_dev = train_test_split(X_train, t_train,
                                                    test_size=0.1, random_state=42)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
X_dev = sc.transform(X_dev)
```

The models was tuned by selecting parameters based from their respective sklearn-wiki page[2][3][4]

- For the logistic regression, the only parameter explored is $C$, selecting values between [1, 10, 100, 1000,10000]
- For the support vector machine the choice fell on the parameters *kernel* ['rbf', 'sigmoid'], $C$ [1, 10, 100, 1000] and *gamma* [1e-3, 1e-4,1e-5]
- For the Neural Network *hidden_ layers_ size* [(100, 100), (200, 100),(200,200),(300,150)], *alpha* [1e-2, 1e-3, 1e-4], and *activation* ['relu','tanh']

The search for the optimal parameters for each model has been conducted using the *GridSearchCV* method. Specifically, the attribute *scoring* was set to *f1_ weighted* due to the multi class nature of the problem, and the number of fold was set to 8.

Both neural network and SVM obtained similar F1 scores on the development set. Although the SVM sometimes obtained better values, after merging the training and development sets, and evaluating in the test set, the neural network created with the best parameters consistently performed better.
Consequently, another evaluation metric was considerate. Keras was used to compute the Softmax Loss Function as the task involves a multi class classification. When comparing the values of the loss,despite the F1 scores on the development set being similar( And sometimes more favorable for the SVM ) the neural network exhibited consistently lower loss values.

This procedure was repeated also on the test set, where the neural network again achieved best performance, both in terms of F1 score and softmax loss. In the figure2.1 the classification report scores and the corresponding Softmax loss values from a run of the script are presented, both for the development and test set. (These values are obtained from the Psychological State Prediction problem)

```
Classification report for SVM on Dev set:              precision    recall  f1-score   support

           0       0.32      0.29      0.30        21
           1       0.41      0.41      0.41        22
           2       0.31      0.25      0.28        20
           3       0.26      0.35      0.30        17

    accuracy                           0.33        80
   macro avg       0.32      0.32      0.32        80
weighted avg       0.33      0.33      0.32        80

Classification report for NN on Dev set:              precision    recall  f1-score   support

           0       0.29      0.29      0.29        21
           1       0.21      0.18      0.20        22
           2       0.31      0.20      0.24        20
           3       0.26      0.41      0.32        17

    accuracy                           0.26        80
   macro avg       0.27      0.27      0.26        80
weighted avg       0.26      0.26      0.26        80

VALUE OF CROSS ENTROPY LOSS FOR DEV SET FOR SVM:  848.0886
VALUE OF CROSS ENTROPY LOSS FOR DEV SET FOR NN:  821.9366
```

```
CLASSIFICATION REPORT FOR NN               precision    recall  f1-score   support

           0       0.33      0.35      0.34        46
           1       0.27      0.27      0.27        51
           2       0.30      0.30      0.30        53
           3       0.24      0.22      0.23        50

    accuracy                           0.28       200
   macro avg       0.28      0.29      0.28       200
weighted avg       0.28      0.28      0.28       200

VALUE OF CROSS ENTROPY LOSS FOR NN:  2285.5747
CLASSIFICATION REPORT FOR SVM               precision    recall  f1-score   support

           0       0.23      0.39      0.29        46
           1       0.27      0.27      0.27        51
           2       0.27      0.21      0.23        53
           3       0.21      0.12      0.15        50

    accuracy                           0.24       200
   macro avg       0.25      0.25      0.24       200
weighted avg       0.25      0.24      0.24       200

VALUE OF CROSS ENTROPY LOSS FOR SVM:  2952.9966
```

Figure 2.1: SVM and NN scores

For this reason, for both tasks, the model chosen was a Neural network, with the following configuration of parameters:

For the Psychological state:

- activation = relu
- alpha = 0.0001
- hidden_layer_sizes = (100,100)
- solver = adam

For the Cognitive load:

- activation = relu
- alpha = 0.01
- hidden_layer_sizes = (300,150)
- solver = adam

After re-merging the train and development test, the final model was created using the optimal parameters found and fitted to the new train set.

# 3. Test and final considerations

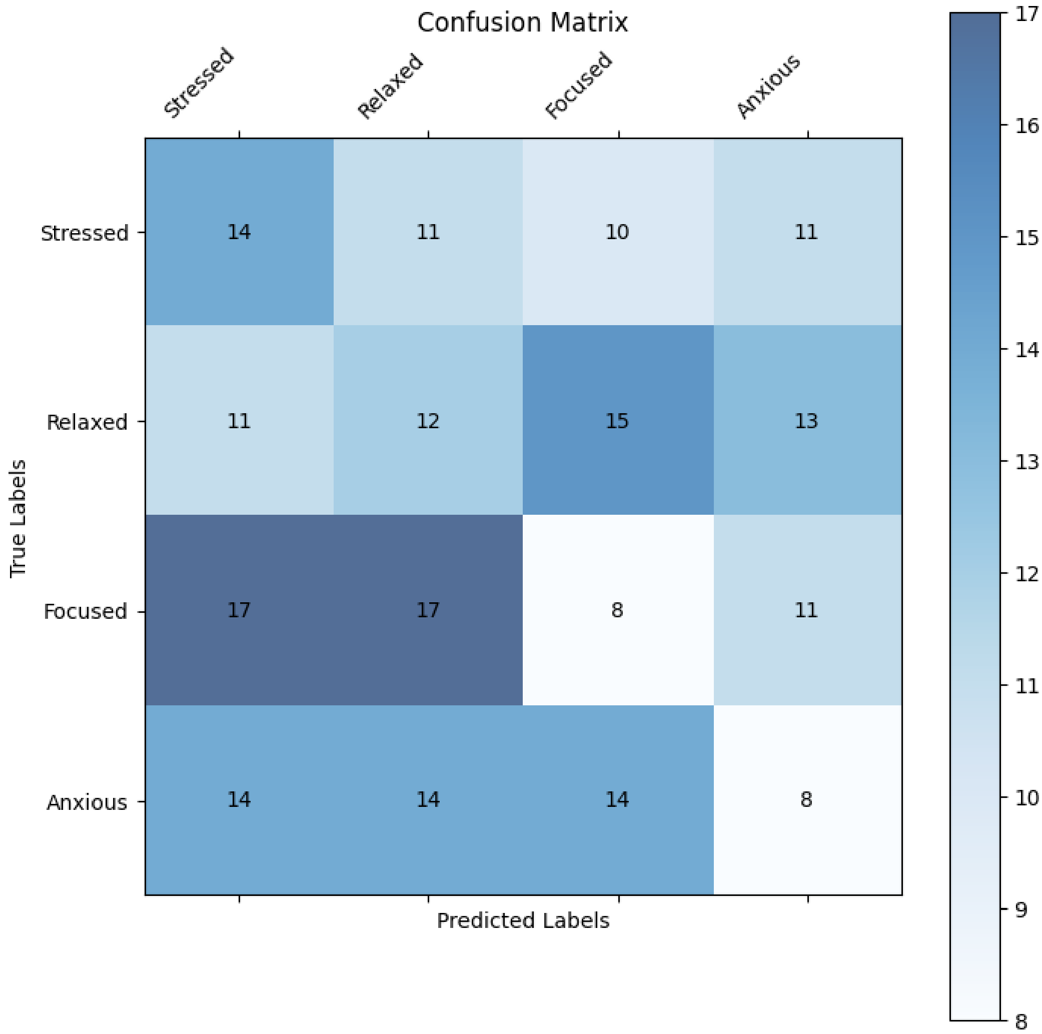
Finally, class prediction was made on the test set.

```
#FINAL MODEL
nn = MLPClassifier(activation='relu', alpha=0.0001,hidden_layer_sizes=(100,100),solver='adam')
nn.fit(X_train,t_train)

t_hat_test = nn.predict(X_test)
print(classification_report(t_test,t_hat_test))
```

```
#FINAL MODEL
nn = MLPClassifier(activation='relu', alpha=0.01,hidden_layer_sizes=(300,150), solver='adam')
nn.fit(X_train,t_train)

t_hat_test = nn.predict(X_test)
print(classification_report(t_test,t_hat_test))
```

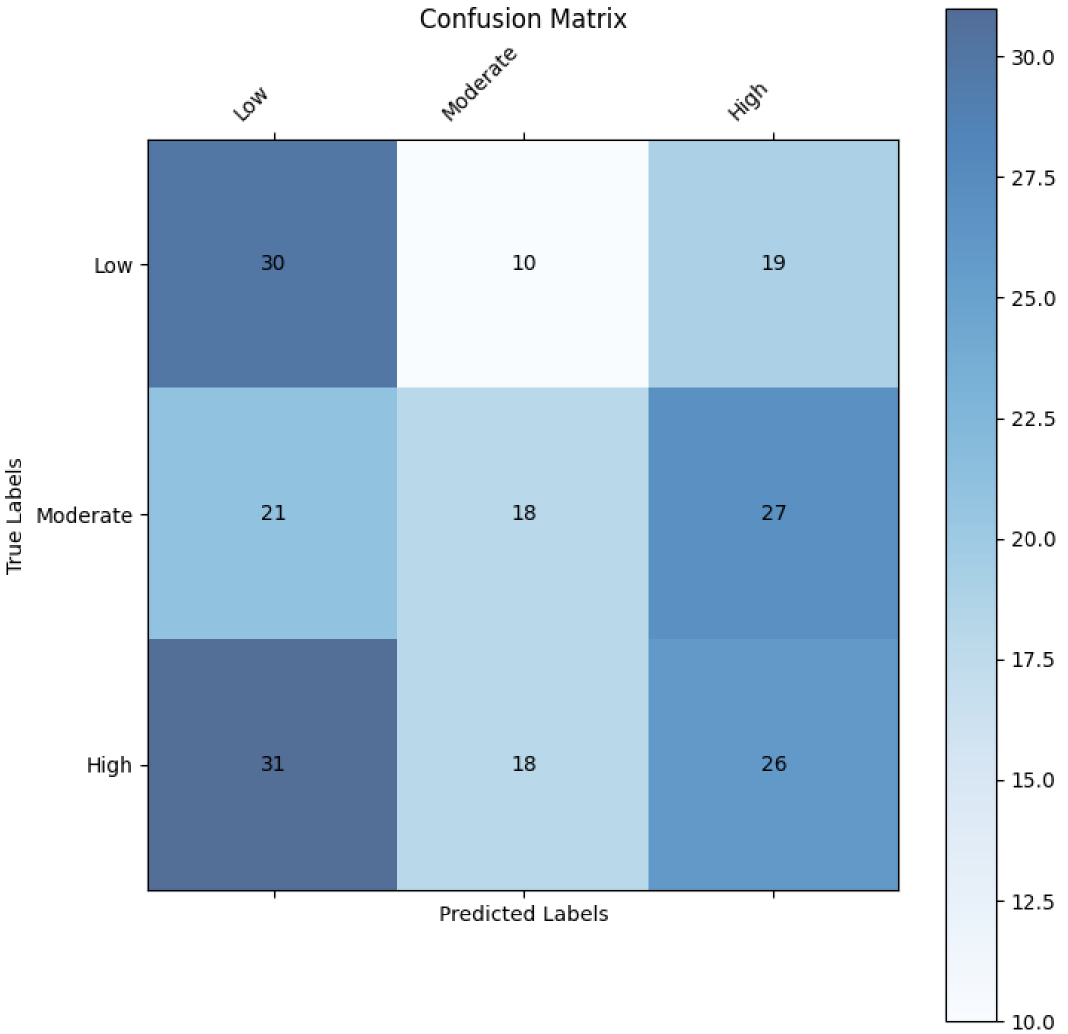
Unfortunately, in both problems, the models were unable to correctly address the classes. In the cognitive load prediction the situation the performance were slightly better, but the F1 scores for the classes still remained below the value of 0.5.

The light improvement in the score may be attributed by the inferior number of classes in the target variable in the Cognitive Load prediction task.




```
CLASSIFICATION REPORT FOR NN                   precision    recall  f1-score   support

           0       0.25      0.30      0.27        46
           1       0.22      0.24      0.23        51
           2       0.17      0.15      0.16        53
           3       0.19      0.16      0.17        50

    accuracy                           0.21       200
   macro avg       0.21      0.21      0.21       200
weighted avg       0.21      0.21      0.21       200

VALUE OF CROSS ENTROPY LOSS FOR NN:  2524.1404
```

Figure 3.1: Scores for Psychological state classification




```
CLASSIFICATION REPORT FOR NN                precision    recall  f1-score   support

           0       0.37      0.51      0.43        59
           1       0.39      0.27      0.32        66
           2       0.36      0.35      0.35        75

    accuracy                           0.37       200
   macro avg       0.37      0.38      0.37       200
weighted avg       0.37      0.37      0.36       200

VALUE OF CROSS ENTROPY LOSS FOR NN:  1980.8975
```

Figure 3.2: Scores for Cognitive load level classification

# 3. Bibliography


[1] *Dataset.* URL: https://www.kaggle.com/datasets/ziya07/psychological-state-identification-dataset/data.

[2] *Logistic regression.* URL: https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html.

[3] *SVM.* URL: https://scikit-learn.org/dev/modules/generated/sklearn.svm.SVC.html.

[4] *Neural network.* URL: https://scikit-learn.org/1.5/modules/neural_networks_supervised.html.