# Milestone 4 – Classify new dataset using the pre-trained MLP from milestone MS3, Confusion matrix and dataset#3.

(Parameter setting and reliability test of a sensor system for infant carrier car seat sensing in a car using a dashboard sensor)

Masters of Engineering
Information Technology

Aneeta Antony(1431461)
aneeta.antony@stud.fra-uas.de

Sonam Priya (1427129)
sonam.priya@stud.fra-uas.de

Nitu Shrestha (1428296)
nitu.shrestha@stud.fra-uas.de

## I. IMPLEMENTATION -MILESTONE 4

### A. *Conduct experiments with and without Baby Doll in parkinglot with RedPitaya dashboardsensor.*

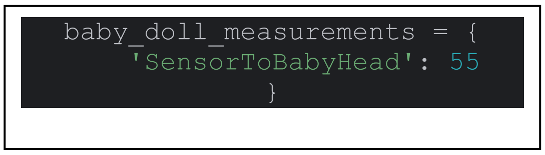
Conduct another set of data measurements and classify them using the pre-trained MLP from milestone MS3. We will submit the data set #3 and the confusion matrix.

This Obtained experiment ADC data is now transformed to fft data using appropriate python libraries and coding techniques.

*1) Convert data set to fft with aditional Label ´Measurement to Baby doll head´ for better classification*

The data set #3 is converted to fft using NumPy inbuilt functions. And this is saved as NumPy data frame array for further use like training the MLP etc. During the conversion we appropriately label the data as 1 or 0, for babydoll presence detected and no babydoll respectively.

Additionally, one measurement label is added to the fft data frame array which was also used in training the MLP, as shown in figure one for the corresponding data readings.



```
    baby_doll_measurements = {
        'SensorToBabyHead': 55
                          }
```

**Figure 1**

This additional label together with the ffft magnitude label contributes to the accuracy of classification and prediction of the trained MLP.

*2) The Trained MLP and Random Forest Classifer are like below.*

```
mlp_classifier =
MLPClassifier(hidden_layer_sizes=(4,),
activation='logistic', solver='sgd', alpha=0.01,
                batch_size='auto',
learning_rate='constant', learning_rate_init=0.01,
                max_iter=100, shuffle=True,
random_state=42, tol=1e-3, verbose=False,
                early_stopping=False,
validation_fraction=0.1)

# Fit the classifier to your data
mlp_classifier.fit(X_train, y_train)

# Predict on test data
y_pred1 = mlp_classifier.predict(X_test)
# Calculate evaluation metrics
accuracy1 = accuracy_score(y_test, y_pred1)
precision1 = precision_score(y_test, y_pred1)
recall1 = recall_score(y_test, y_pred1)
f1_1 = f1_score(y_test, y_pred1)
```

```
param_grid = {
   'n_estimators': [10, 50, 100],  # Number of trees in the
forest
   'max_depth': [None, 5, 10],  # Maximum depth of the
tree
   'min_samples_split': [2, 5, 10],  # Minimum number
of samples required to split an internal node
   'min_samples_leaf': [2, 4, 6]  # Minimum number of
samples required to be at a leaf node
}
# Initialize RandomForestClassifier
classifier = RandomForestClassifier(random_state=42)
# Perform grid search with cross-validation
grid_search = GridSearchCV(estimator=classifier,
param_grid=param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
# Get the best parameters and best estimator
best_params = grid_search.best_params_
best_estimator = grid_search.best_estimator_
# Evaluate the best model on the test set
y_pred = best_estimator.predict(X_test)
```

## B. *Saved MLP CLassifier and RFClassifier.*

*1) After training the MLP Classifer and RFClassifier for further use.*

```
# Save the mlp classifier to a file
joblib.dump(mlp_classifier,
'mlpClassifier_model_trained.pkl')

# Save the best_estimator to a file
joblib.dump(best_estimator,
'random_forest_model_trained.pkl')
```

*2) Reload the MLP on the dataset#3*

```
# Load the saved model
Loaded_mlp =
load('mlpClassifier_model_trained.p
kl')

# Load the saved model
Loaded_mlp =
load(random_forest_model_trained.pk
l)
```

## C. *Confusion Matrix created from trained MLClassifier and RF Classifier*

Toward this milestone we generated a confusion matrix from the MLPs created. Comparing the predicted labels with test labels. Predicted label and accuracy are found for the dataset#3 test data.

*1) The Confusion Matrix is obtained form the trained MLPClassifer in Figure 2, show between "With Baby" and "Without Baby."*

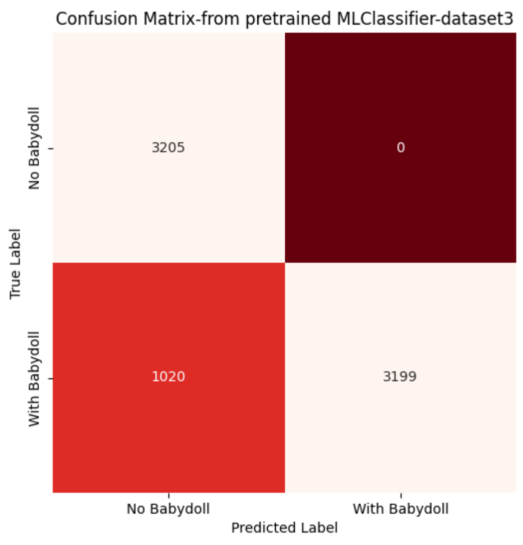


**Figure 2 Confusion Matrix**

*2) The Confusion Matrix is obtained form the pre trained RF Classiffer model in Figure , show between "With Baby" and "Without Baby."*

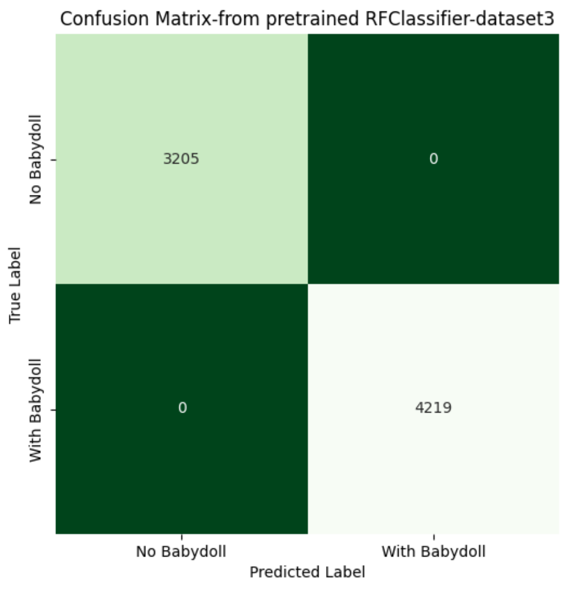


**Figure 3 Confusion Matrix**

*3) The Hyperparameter modified for the MLClassifer*

*a) In the hidden layers, increase the number of neurons (hidden_layer_sizes).*

*b) Tried out various optimization methods (solvers), such as "lbfgs," "sgd," and "adam."*

*c) Experimented with other activation functions, such as sigmoid, tanh, and relu.*

*d) To avoid overfitting, enabled early stopping (early_stopping=True). Training will end when performance on a validation set no longer improves.*

*e) To enable the model to converge, varied the maximum number of iterations (max_iter).*

*4) Data Preprocessing to have extra label for better classification and Train.*

During Data Preprocessing window used appropriate window function and sampling rate in order to properly form the data set. Properly labeling the data according to the e knowledge for supervised experiments. Additional Label like measurement parameters were added to improve classification and training of the models. ('Frequency', 'FFTMagnitude','Phase', ´SensorToBabyHead´)

### *D. Dataset#3 , FFT Dataset#3.*

The dataset#3 both ADC Data and Processed transformed fft can be found in the OneDrive Link - Dataset3
(FFT Data_data folder and ADC Data Folders) .The fft data is processed and saved as numpy dataframe arrays are saved for the with-baby-doll and with-out-baby- doll namely withbaby_npy_array_DS§ & withoutbaby_npy_array_DS3 .

## REFERENCES

[1] K. S. Gill, V. Anand, R. Gupta and P. -A. Hsiung, "Detection of Malware Using Machine Learning techniques on Random Forest, Decision Tree, KNeighbour, AdaBoost, SGD, ExtraTrees and GaussianNB Classifier," 2023 4th IEEE Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2023, pp. 1-7, doi: 10.1109/GCAT59970.2023.10353495. keywords: {Codes;Digital systems;Computational modeling;Malware;Stability analysis;Data models;Pattern recognition;Data Science;Behaviour Analysis;Malware Attacks;Anomaly Detection;Classifcation;Machine Learning;Deep Learning;Artifcial Intelligence;Classifers}

[2] K.N. Fang, J.B. Wu, J.P. Zhu and B.C. Xie, "A review of technologies on random forests", *Statistics and Information Forum*, vol. A26, pp. 32-38, 2011.

[3] D. Yuan, J. Huang, X. Yang and J. Cui, "Improved random forest classification approach based on hybrid clustering selection," 2020 Chinese Automation Congress (CAC), Shanghai, China, 2020, pp. 1559-1563, doi: 10.1109/CAC51589.2020.9326711. keywords: {Random forests;Decision trees;Clustering algorithms;Classification algorithms;Indexes;Vegetation;Partitioning algorithms;random forest algorithm;clustering ensemble selection;personal indoor thermal preference}

[4] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html