Faculty of Engineering
Cairo University

Systems & Biomedical Engineering Department

Artificial Intelligence in Medicine - Final Project

Human activity recognition

Submitted to:
Dr. Inas Yassin
Eng. Merna Bibars

Submitted by:

Name	Sec	BN
Andrew Muhsen	1	15
Habiba Ahmed	1	23
Tarek Salah	1	44
Muhammed Alaa	1	17

Cairo University
Please let us know if you need to provide any further details.

Andrew.naaem99@eng-st.cu.edu.eg
mohamedalaa20050@gmail.com

Introduction

In numerous healthcare applications, activity Recognition is a critical duty. It is feasible to provide automatic suggestions to both patients and physicians by continually monitoring and analyzing user behavior.

Smartphones and smartwatches are in everyone's hands these days, and such devices come with sensors (Inertial Measurement Units, Temperature sensors, optical sensors) that collect data and tell us information about our bodies' conditions. As technology advances, it becomes more advantageous to use fewer sensors while obtaining a reasonable amount of information.

Our project aims to determine whether we can use fewer sensors to acquire useful information, what machine learning model might be deployed, and what should be done to improve our classifiers' performance.

PAMAP2

The PAMAP2 Physical Activity Monitoring Dataset is an ML dataset from the UCI repository that contains data from 18 different physical activities conducted by 9 participants while wearing three inertial measurement units and a heart rate monitor.

1st Part - Data Processing

1. **Data nature**
2. **Activities selection**
3. **Handling Missing Data**
4. **Handling class imbalance I**
5. **Features selection**
6. **Handling class imbalance II**
7. **Create Modeling Data**
8. **Data visualization**

1.1 - Data Nature

Because this dataset is realistic, some data is missing. Missing data is a common occurrence in data analysis, and it can be caused by a variety of factors, including measuring device malfunction and data collecting mistakes.

In our scenario, the reasons for missing data are as follows:

- Data dropping due to using wireless sensors
- Problems with the hardware setup, causing e.g. connection loss or system crash
- Missing data duo, the difference in sampling frequencies on devices that the IMUs have a sampling frequency of 100Hz and the sampling frequency of the Heart Rate monitor was 9Hz

1.2 Activities selection

PAMAP2 contains data from 18 different physical activities conducted by 9 participants, so it has 18 different IDs, each representing a specific activity (such as walking, cycling, playing soccer, etc.)
Eight subjects' data were taken instead of 9, as the 9th subject didn't do any of the activities we have chosen and optional data were ignored.
Only 4 activities were chosen to predict from.

- **|| *ID 1: lying* ||:** lying quietly while doing nothing,
- **|| ID 2: sitting ||:** sitting in a chair
- **|| ID 4: walking ||:**
- **|| ID 5: running ||:**

1.3 Handling Missing Data

We purposed two solutions for our missing data problem.

- The missing sensory data due to wireless data dropping and hardware problems indicated with NaN were dropped from data, dropping a few rows in data such as PAMAP2 won't affect it much. ***[implemented after handling classes imbalance]***
- the missing Heart Rate values are also indicated with NaN and were solved by linear interpolation for each participant

It was okay to set all NaN values that were resulted because of data dropping or hardware problems as they are not that frequent, and they don't form a fraction of our data set, on the other hand, the missing HR values represented 90% of the

Heart Rate feature so introducing Linear interpolation was a great and reasonable solution, the results of our interpolation method, Fig (1a) & Fig (1b) show the null values before and after interpolation respectively, Fig (2a) & Fig (2b) show how Heart-rate values changed from null to real values by linear interpolation.

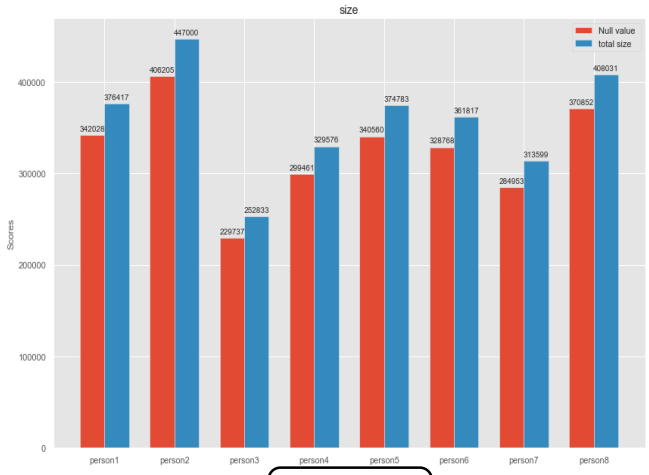


Fig (1a)

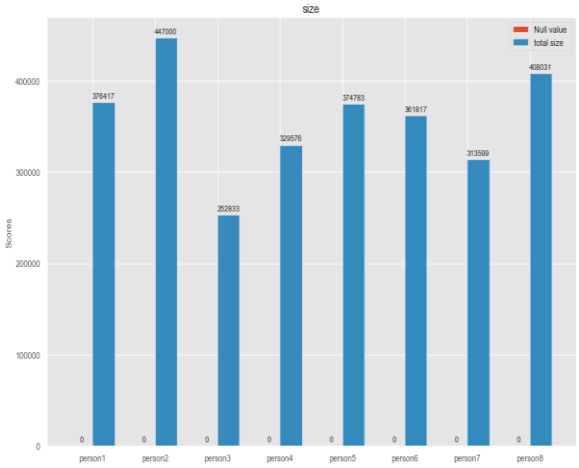


Fig (1b)

persons[1]

✓ 0.2s

	timestamp	activityID	heart_rate	IMU_hand_temperature
0	5.64	0.0	NaN	33.000
1	5.65	0.0	NaN	33.000
2	5.66	0.0	100.0	33.000
3	5.67	0.0	NaN	33.000
4	5.68	0.0	NaN	33.000
...	...	...	...	...
446995	4475.59	0.0	NaN	29.125
446996	4475.60	0.0	NaN	29.125
446997	4475.61	0.0	NaN	29.125
446998	4475.62	0.0	NaN	29.125
446999	4475.63	0.0	134.0	29.125

447000 rows × 55 columns

Fig (2a)

persons[1]

✓ 0.1s

	timestamp	activityID	heart_rate	IMU_hand_temperature
0	5.64	0.0	100.0	33.000
1	5.65	0.0	100.0	33.000
2	5.66	0.0	100.0	33.000
3	5.67	0.0	100.0	33.000
4	5.68	0.0	100.0	33.000
...	...	...	...	...
446995	4475.59	0.0	134.0	29.125
446996	4475.60	0.0	134.0	29.125
446997	4475.61	0.0	134.0	29.125
446998	4475.62	0.0	134.0	29.125
446999	4475.63	0.0	134.0	29.125

447000 rows × 55 columns

Fig (2b)

1.4 Handling class imbalance I

Taking a look into the data some Activities took much more time than others, therefore, it has more data than other activities which are okay for data collecting, but

during splitting into test-train portions to fit models, models will always be biased to the class [Activity] which have more data in the training set.

To solve this problem, to perform a population generalize for the data

- data was split into train and test data sets for every person independently and concatenated so that every subject has the same ratio in the training and testing data
    - Fig (3a) & Fig (3b) show ratios of activities for each subject in both training and testing sets.

Fig (3a) - Training

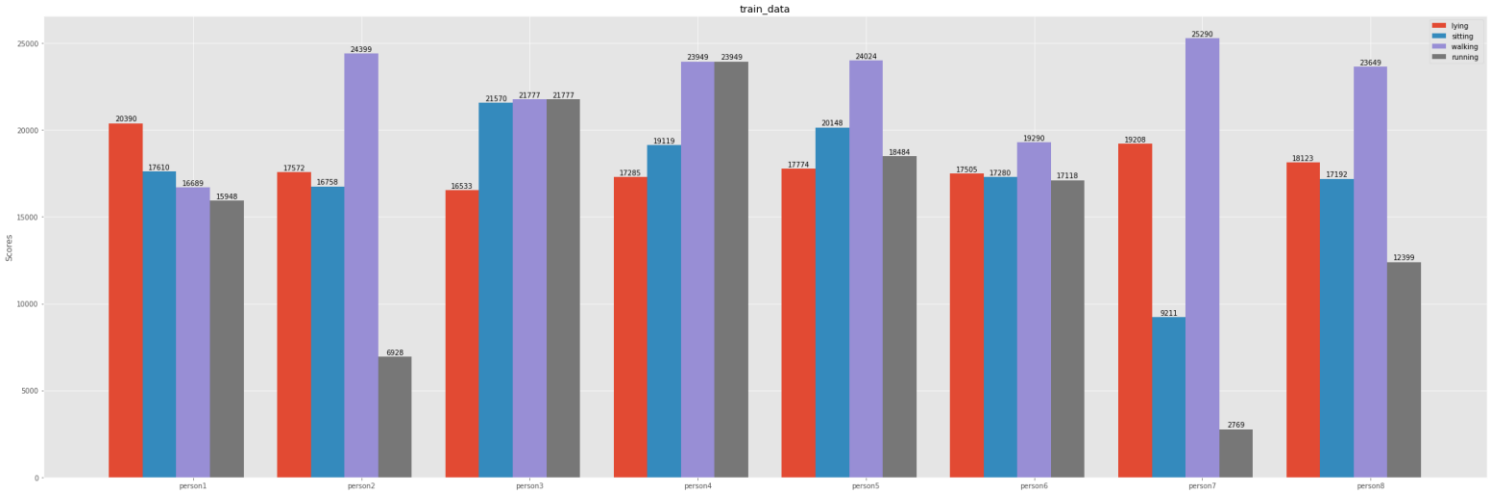
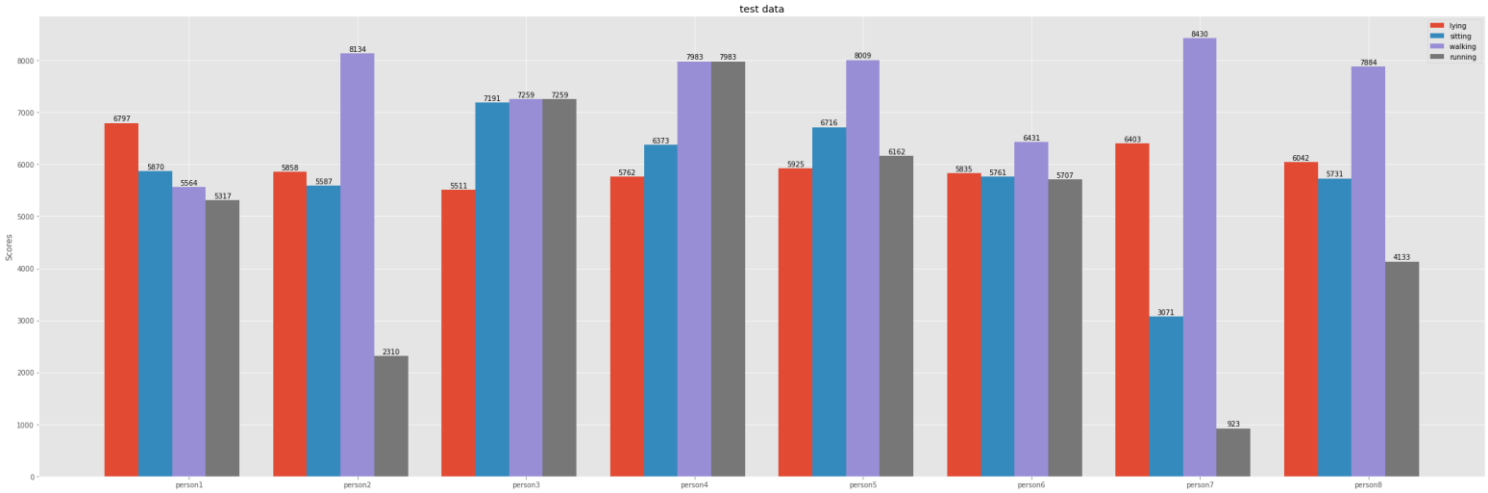


Fig (3b) - Testing



- Data was also split for every activity independently and concatenated so that every activity has the same ratio in the training and testing data.
    - Fig (4a) & Fig (4b) show ratios of activities in both training and testing sets.

Fig (4a)

Fig (4b)

    - After that data were **shuffled**, to make sure that data is not organized by subject number

1.5 Features selection

To select the best features, we used 3 methods

- Dropped useless features by just looking at the data
- Plotted Heat-map for features correlations, then dropped redundant features
    - Fig (5a) shows the plotted Heat-map and how it showed the redundant features that gave the same information

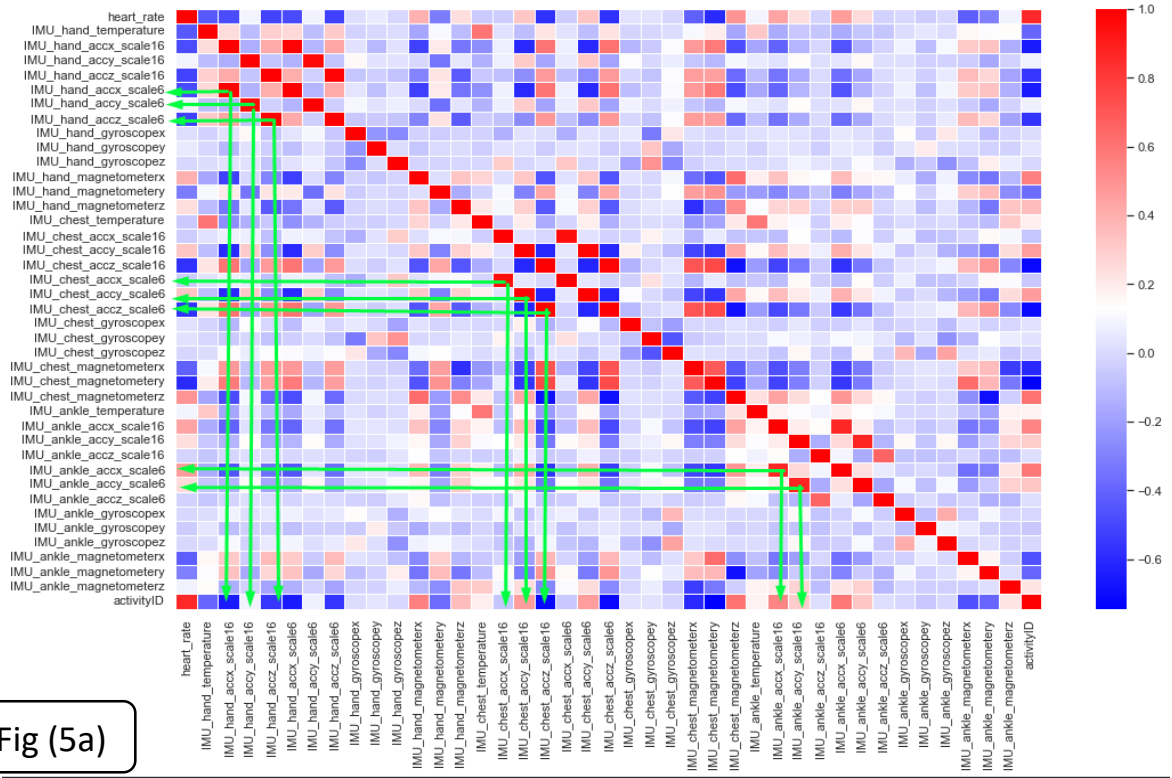


Fig (5a)

- Checking feature ratings and deciding which features to exclude using Sklearn SelectKBest
    - The SelectKBest method selects the features according to the k highest score. Can be applied for both classification and regression data
    - Fig (6a) shows the output of the SelectKBest function, features scores
    - Fig (6b) shows a pie chart for the selected features for modeling

	feature_name	score_value
0	heart_rate	0.375794
14	IMU_chest_accz_scale16	0.145659
2	IMU_hand_accx_scale16	0.073358
8	IMU_hand_magnetometerx	0.061212
18	IMU_chest_magnetometerz	0.051328
13	IMU_chest_accy_scale16	0.043507
20	IMU_ankle_accx_scale16	0.040198
11	IMU_chest_temperature	0.037406
4	IMU_hand_accz_scale16	0.032929
9	IMU_hand_magnetometery	0.023353
26	IMU_ankle_magnetometerx	0.022159
1	IMU_hand_temperature	0.019721
28	IMU_ankle_magnetometerz	0.017979
27	IMU_ankle_magnetometery	0.016930
10	IMU_hand_magnetometerz	0.014084
21	IMU_ankle_accy_scale16	0.009909

Fig (6a)

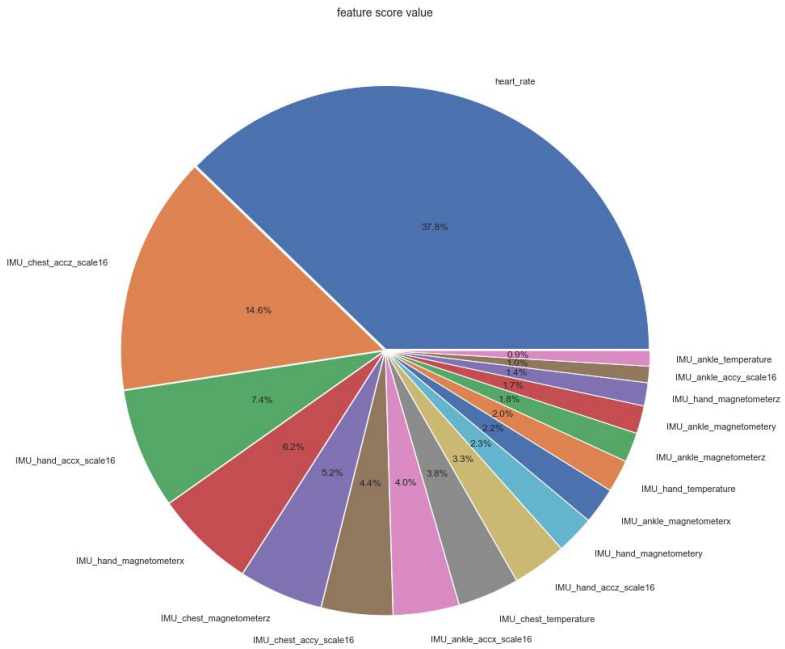


Fig (6b)

1.6 Handling class imbalance II

Imbalanced Data Distribution is a concept used frequently in Machine Learning and Data Science to describe when observations in one class are significantly higher or lower than observations in other classes. Machine Learning algorithms don't consider class distribution since their goal is to improve accuracy by minimizing error. This issue is common in areas such as fraud detection, anomaly detection, and facial recognition.

We tried a few approaches but none of them worked. Our answer was to chop down some data from the majority of classes to balance the classes in a set called Balanced data.

- SMOTE
- Near Miss Algorithm

So we created a balanced data set for each of the training and the testing sets using the shuffling & drop technique

- We set some threshold for each class data
- Dropped excessive data
- Used the **shuffle** function to assure having data from all 8 subjects
- Creating a balanced version of train and test sets
- Fig (7a) shows classes' distribution of training and testing data sets after using our technique

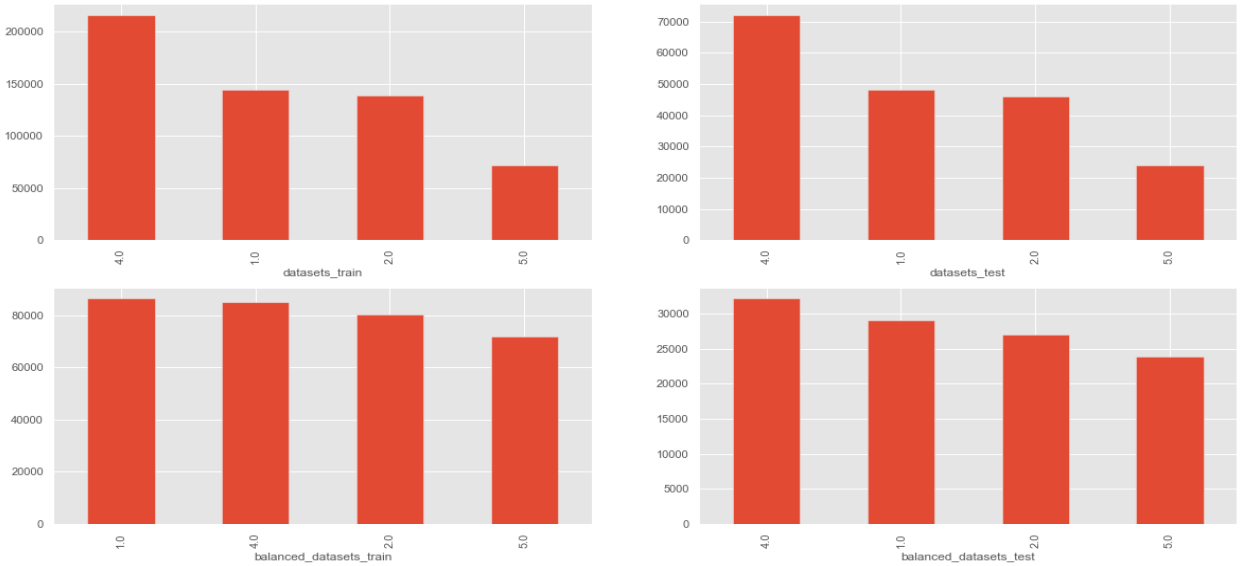


Fig (7a)

- Fig (7b) shows subjects' data distribution in training and testing data sets after using our technique

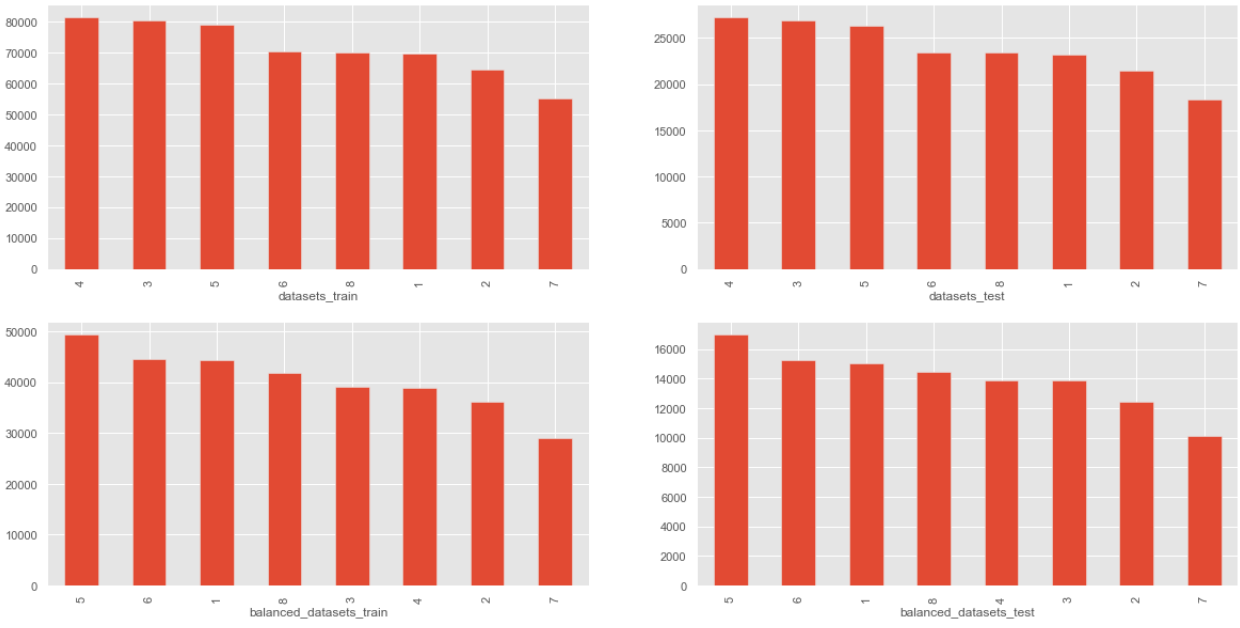


Fig (7b)

1.7 Create Modeling Data

We picked a piece of the balanced data set that has all of the balanced data features and distribution but is substantially lower in size. Fig (8a) & Fig (8b)

The figures below validate that the Mini data set is derived from balanced data.

- Fig (8a) confirms the same classes distributions

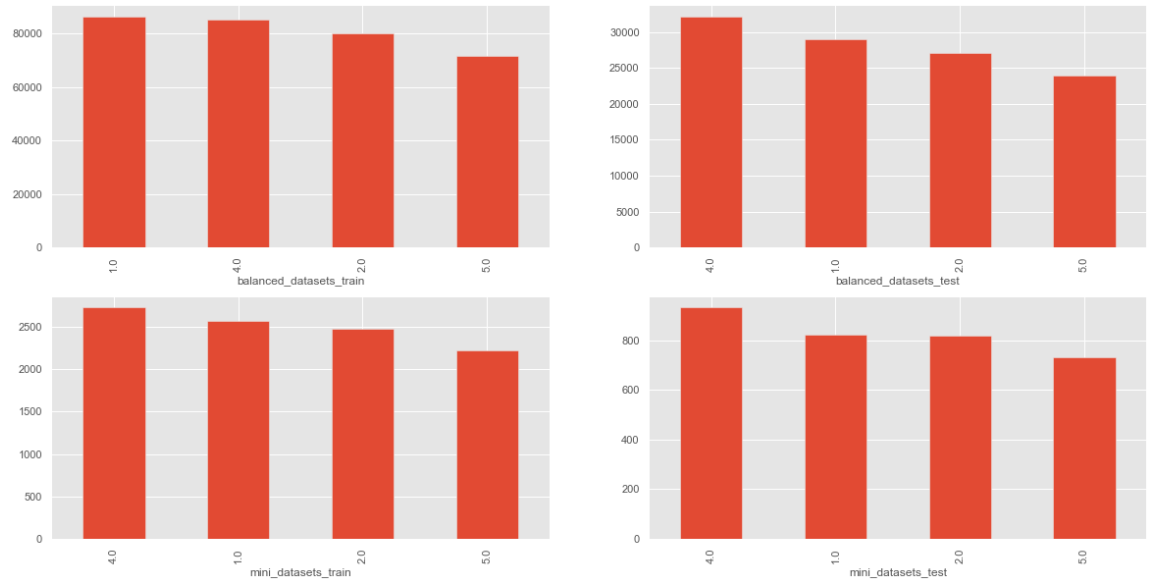


Fig (8a)

- Fig (8b) confirms the same subjects distributions

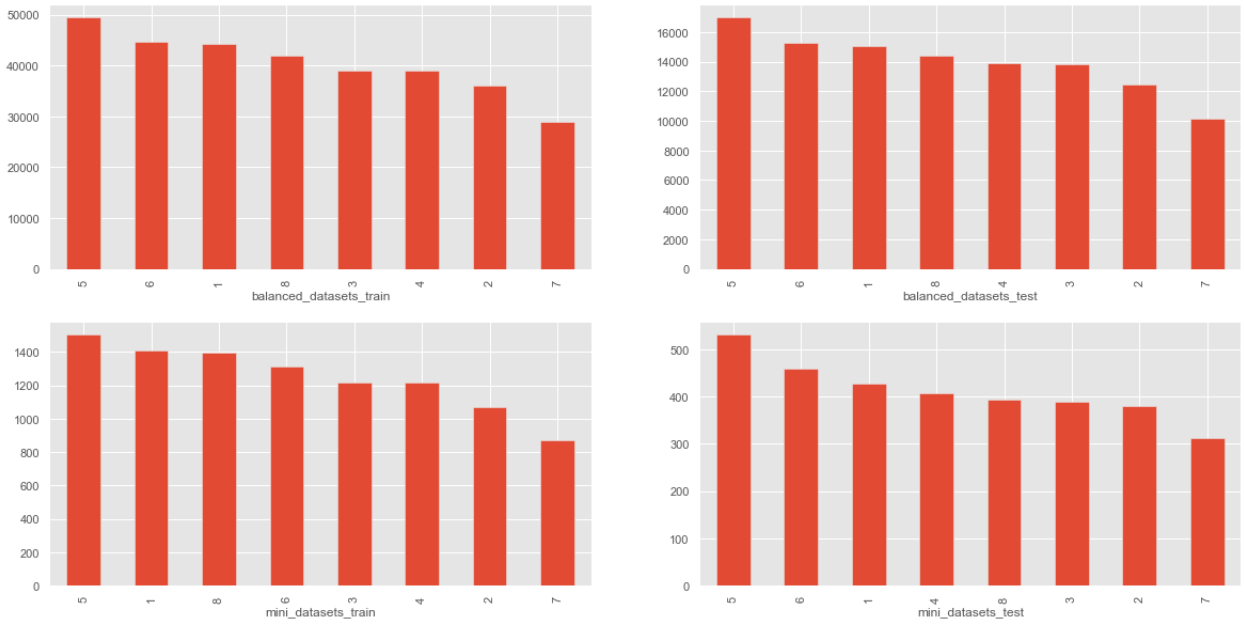


Fig (8b)

1.8 Data Visualization

Finally, we implemented PCA to reduce our data dimensions to two so that we could display data more clearly and examine what we did with our data during the processing step.

Fig (8a) demonstrates our data plot as Original, balanced, and small respectively, with PCA plot on the left and normal, scatter on the right.

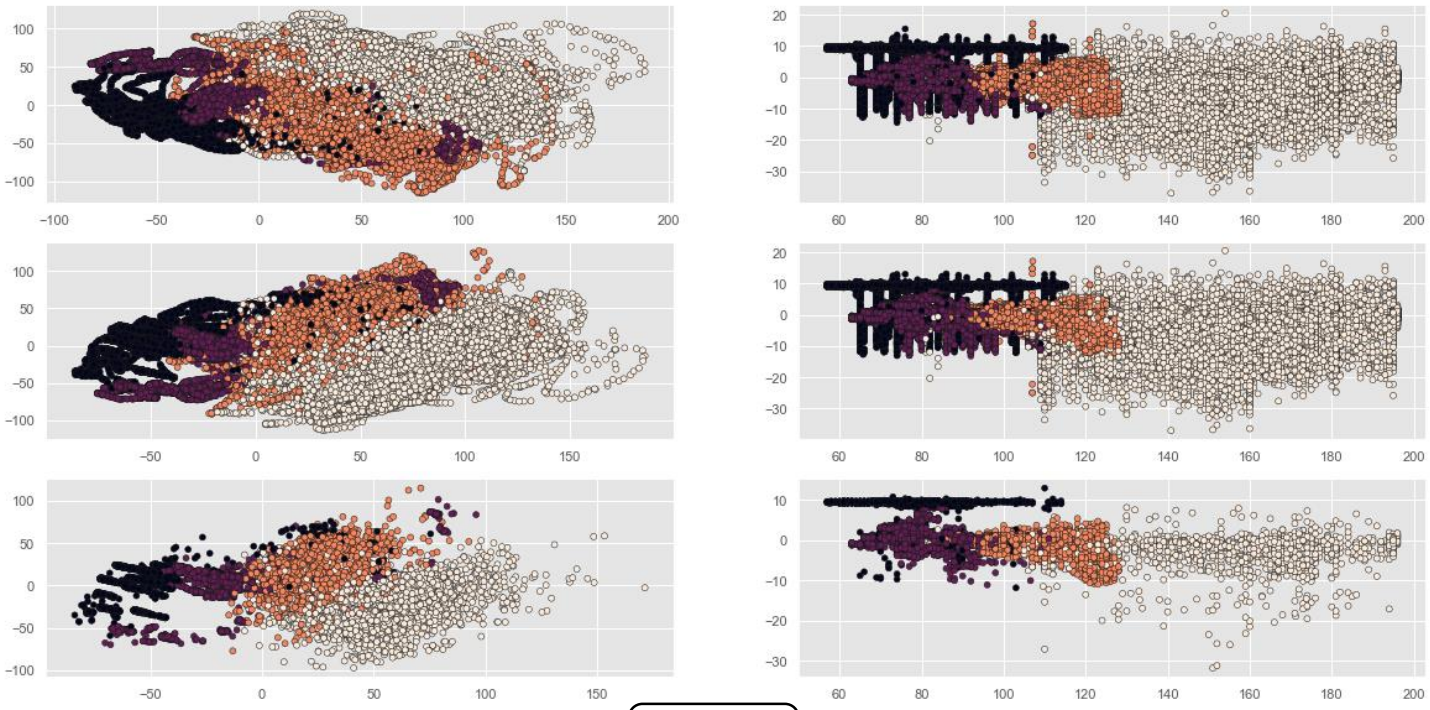


Fig (8a)

After graphing, we determined that the tiny data set is ideal for modeling since it represents PAMAP for four activations. The next step is to include this data set in modeling and observe how well Models perform using only this mini model to estimate the original data class.

We reduced our data from 18 to four classes and from 52 to 16 characteristics.

Let's move on to Part 2: Modeling.

2nd - Part Modeling

1- Choosing models
2- Feature values distribution
3- GridSearch
4- Classification report function
5- Models results
6- Accuracy tabel

2.1 Choosing models

Five distinct models were selected based on what was taught throughout the Course and how simple it was to deal with them using Sklearn, these models were:

- **Logistic Regression**
- **Support Vector Machine (SVM)**
- **Decision tree classifier**
- **Random Forest classifier**
- **K-Nearest Neighbor**

Looking at our data, it's clear that **Logistic Regression** is a great model for such data, after picking the important features, it's predicted to run quickly and with moderate accuracy.

When working with large datasets, the **SVM** is recommended; the model is expected to perform well using the Radial Basis Function (RBF).

After selecting only four classes from a dataset with 16 characteristics to predict, the **Decision Tree Classifier** model was chosen to be trained alongside the **Random Forest Classifier**, which consists of branches of trees operating in parallel.

After balancing the dataset, it is safe to claim that **K-Nearest Neighbor** is anticipated to perform the best.

2.2 Feature values distribution

To work with features that have a wide range of values, a distribution approach is used to overcome feature value indifference and give the feature its true weight in prediction making, two distribution models were used.

- **Normalize (L2)**
- **Standard scaler**

Standard scaler distribution appeared to be significantly slower as data volumes increased, and while it works somewhat slower with certain models, it generally provides superior accuracy when data is not massive.

2.3 GridSearch

We used L2 - Normalized mini data set for fast performance, our parameters are tuned to fit the normalized model more, but they managed to perform well on standardized data. 5 grid searches were implemented on the 5 models to find the best model parameters for our problem, we used L2 - Normalized mini data set for fast performance, our parameters are tuned to fit the normalized model more, but they managed to perform well on the standardized data.

All GridSearches were implemented in a separate notebook to save time, which can be found in **Gridsearch ground.ipynb notebook**, and thereafter models will be recreated with updated parameters in **Modeling&Results.ipynb notebook**.

2.4 Classification_report

We utilized the classification report function to forecast which model would outperform other models before computing the model score on a larger data set, and it indicated that **KNN** will outperform all other models.

2.5 Models Results

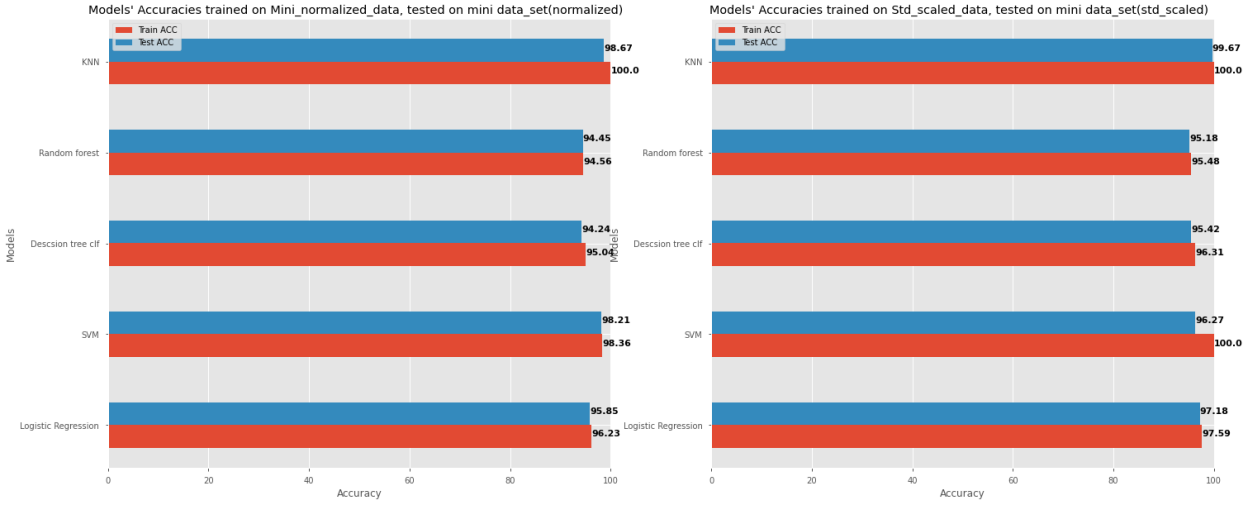
In each phase, a data set was trained on both standardized scaled models and normalized models, and accuracies were shown. The results of each phase will be detailed below each Accuracy graph

It's worth mentioning that phases two and three took a long time to complete because five models were evaluated on both balanced and original data.

The accuracies are excessively high since we preprocessed the data to maximize the potential of all models, by feature selection and class balance, while on the other hand, we maximized models' performance using GridSearch, and changing the distribution of our data.

All models were pushed to their absolute limits.

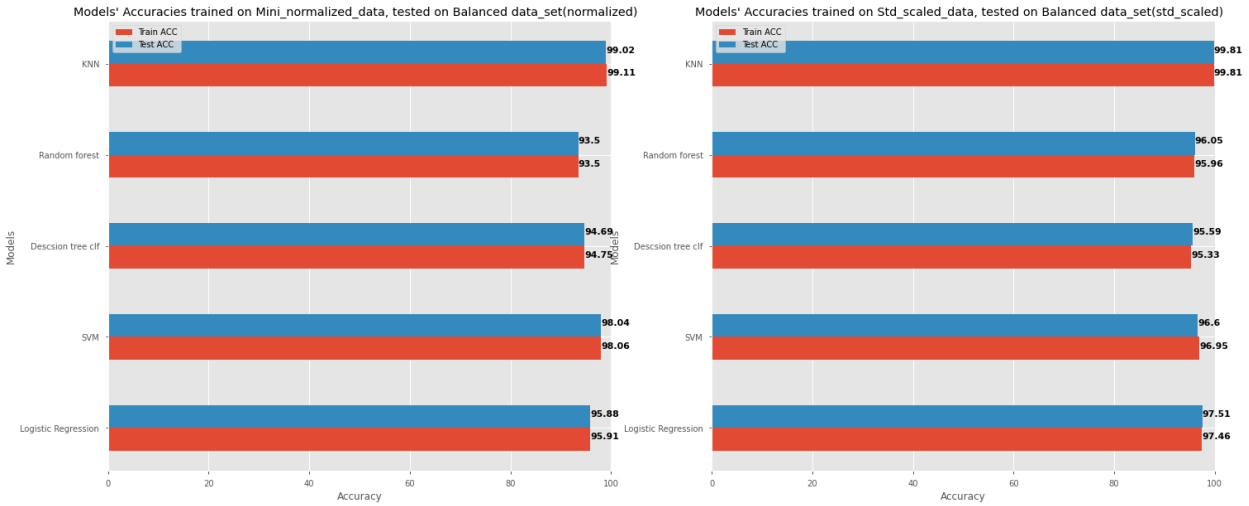
- Phase one, testing the model on the mini sample & results



Results of phase one

- **Performance**
  - Std scaler small sample models performed considerably better than normalized mini sample models, but the processing time was about tripled (time complexity).
  - For rapid fitting and excellent accuracy results, normalized mini sample models were preferable.
- **Notes**
  - The reason for the std scaler's sluggish performance is the fit transform feature, which requires reshaping data to fit the model before calculating the score, which takes longer than the normalization method.
  - Due to the nature of the Original _data set, KNN with n neighbors = 1 appears to be working very well.
- **Conclusions**
  - Logistic Regression is the Fastest model.
  - As expected by the categorization report, KNN has the best accuracy.
  - KNN and SVM appear to function quickly and accurately, but slower than Logistic regression.
  - Logistic regression is the best classifier if the data is normalized since it is extremely quick.

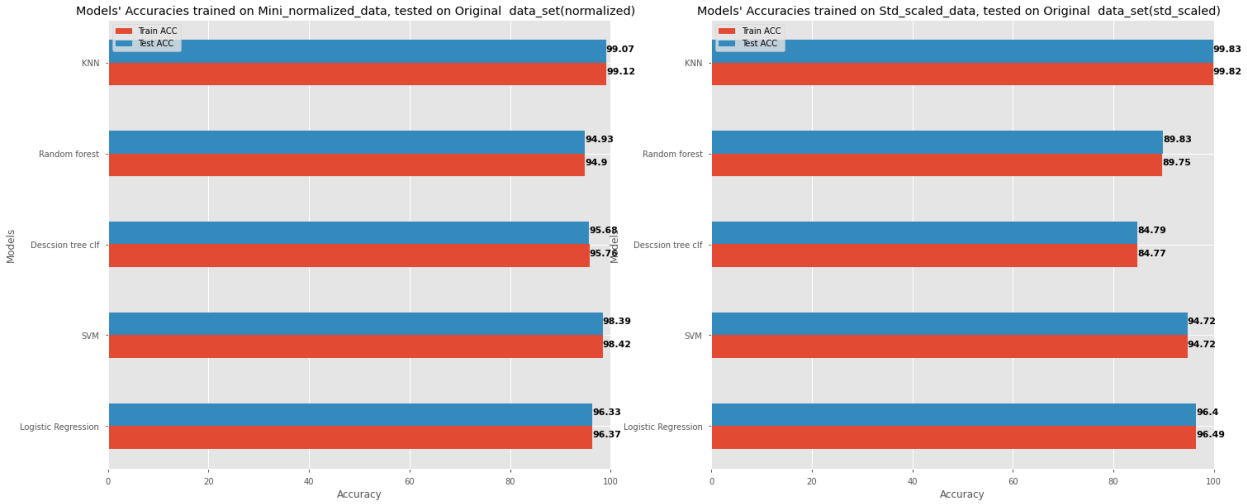
- Phase two, testing the model on balanced data & results.



Results of phase two

- **Performance**
  - std scaler worked somewhat better than predicted.
  - The time complexity of the std scaler model is significantly higher; it took more than three times as long as the normalized model.
- **Notes**
  - the same cause for the sluggish performance
  - Data must be normalized or std_scaled according to the model.
- **Conclusions**
  - KNN is on the lead
  - normalized model : SVM & KNN seems to be the best models to fit on the balanced data
  - Logistic regression is still the fastest ,and with noticeably high accuracy

- Phase three, testing models on Original data & results



Results of phase three

- **Performance**
  - Normalized models outperform std models substantially.
- **Notes**
  - The same cause for the sluggish performance
  - Std_scaler models took roughly 7 times the time Normalized models took
- **Conclusions**
  - As indicated by the classification report, knn has the best accuracy.
  - Normalized models are better than std_scalers models for this Data_set
  - **It's best to use logistic regression,** because it's significantly quicker than other models and has a slightly lower accuracy trade-off.
  - **It's preferable to use KNN,** since it provides somewhat greater accuracy at a lower cost of time complexity.

2.6 Accuracy tabel

Models Results on Normalzied data

Model	Train Accuracy	Test Accuracy
K-Nearest Neighboor	99.99%	98.67%
Random forest classifer	94.56%	94.45%
Descsion tree classifer	95.04%	94.24%
Support vector machine	98.36%	98.21%
Logistic Regresssion	96.23%	95.85%

Models Results on standardized data

Model	Train Accuracy	Test Accuracy
K-Nearest Neighboor	99.82%	99.83%
Random forest classifer	89.75%	89.83%
Descsion tree classifer	84.77%	84.79%
Support vector machine	94.72%	94.72%
Logistic Regresssion	96.49%	96.4%

3rd Conclusion

- As a result of data preprocessing and rigorous modeling adjustments, all models performed with more than 95% accuracy
- Standardizing has a great effect on small-sized data, but normalization has a considerably greater impact on large-sized data.
- Standardized models are generally slow
- KNN has the highest accuracy of all models as the classification report assumed
- Logistic regression is the quickest model for prediction
- It is advised that a Logistic model be used on a normalized data sample to achieve an accurate and quick prediction.
- KNN is a great choice for such data but at the cost of time complexity