Wet section

To succeed in the classification task in the best possible way, we first had to go back into feature imputation stage.

Code Review

As before, our code is fully automatic, and performs the model selection automatically.

To the existing model we have added new models, which are AdaBoost, and the MLP.

The same actions on features are later done on the unseen data, and the output is given.

Features imputation

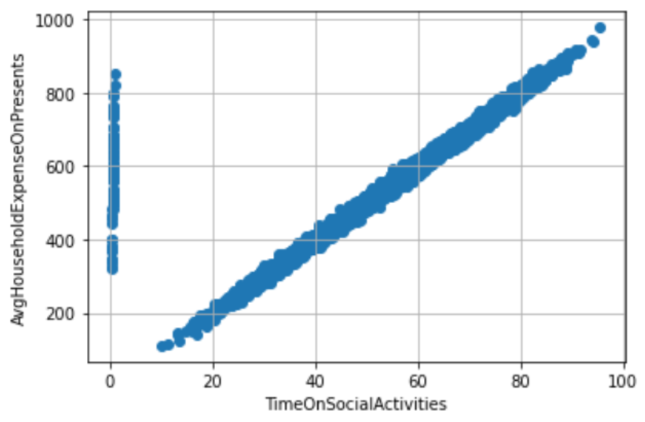
Some feature imputations techniques depend on the dataset, such as replacing missing values with the mean value. For such cases, the mean values found from the operation on the training set were found and saved. And later used on the test set and the unseen data.

1. Discipline score

The discipline score had outliers, but it was noticed that they are simply multiplications of existing discipline scores by 10. Those outliers were divided by 10. The other missing were replaced with a mean.

2. Social Activities

There is a big correlation between the TimeOnSocialActivities and the AvgHouseholdExpenseOnPresents graphs:

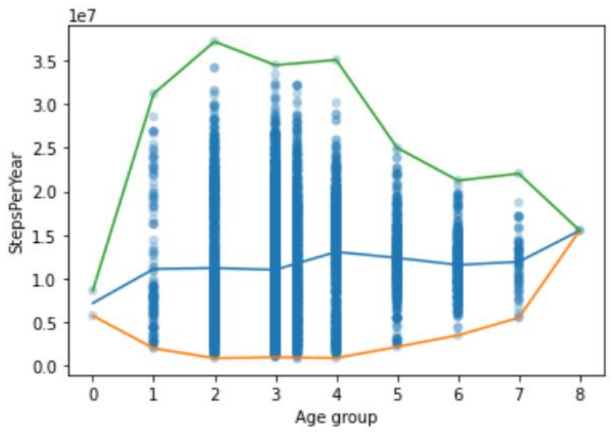


The following imputations are done:

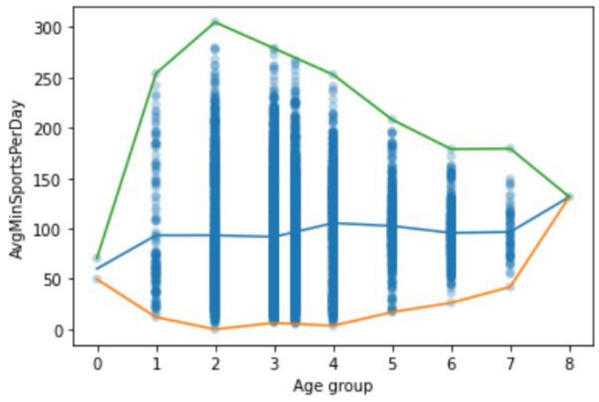
- The outliers (TimeOnSocialActivities < 5) was replaced according to AvgHouseholdExpenseOnPresents
- The non-existing values were also replaced according to AvgHouseholdExpenseOnPresents (if those existed there)
- For all the rest, replacing with the mean value of TimeOnSocialActivities

3. Age Handler

To handle missing age, we have examined the AvgMinSportsPerDay and StepsPerYear parameters, and this is the distribution (with upper and lower boundaries)



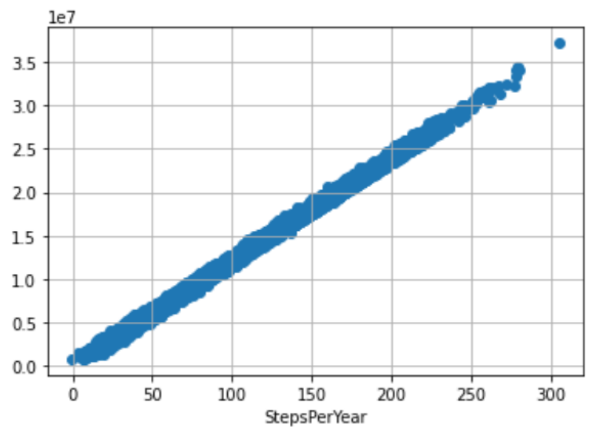
Same for AvgMinSportsPerDay:



We can see some correlation (which is logical).

4. Steps Per Year

There is a strong correlation between AvgMinSportsPerDay and StepsPerYear:



So if the value for StepsPerYear is missing, but is existing in AvgMinSportsPerDay, we calculate this value. Other values are replaced by the mean value.

5. PCR Results

Like with the case with clustering (homework no. 4) we try both the mean and the KNN imputer. For each PCR result, we use the z-value to discover outliers. Then calculate the mean without the outliers. Then we replace the outliers and the missing values by those means or the KNN.

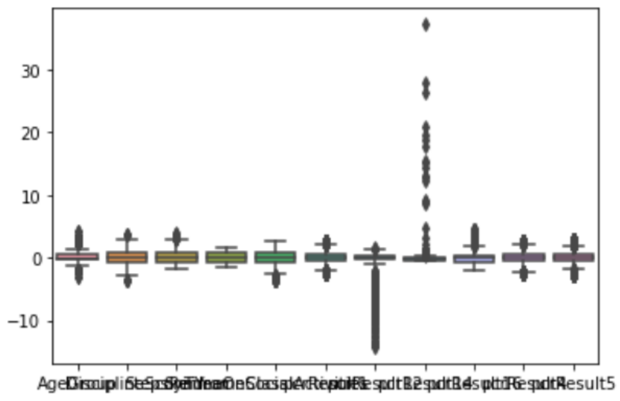
- For training set: we SAVE the mean and STD values that were calculated
- For test/unseen sets: we use the saved mean and std values for each feature to find outliers in the unseen dataset. (we use the same z-score value). We then replace the outliers and missing values with the mean.

6. Syndrome Class

No correlation was found between this feature and any other feature. So we add a new categorical value '5' for 'unknown syndrome class'.

Features Scaling

StandardScaler was used on all features. The boxplot distribution of the data after the scaling:



Cluster Analysis

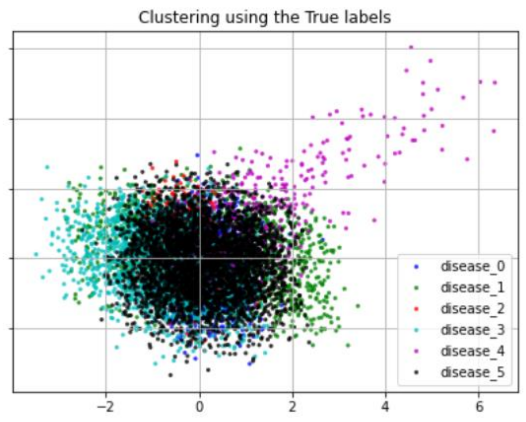
By learning to perform the cluster analysis, we tried to visualize the 6 clusters of the disease type. This would also help us to verify that the feature imputation procedures were indeed effective.

We use the most successful clustering model from the previous homework, which is GaussianMixture Model, since most of the features distributions resemble Gaussian distribution.

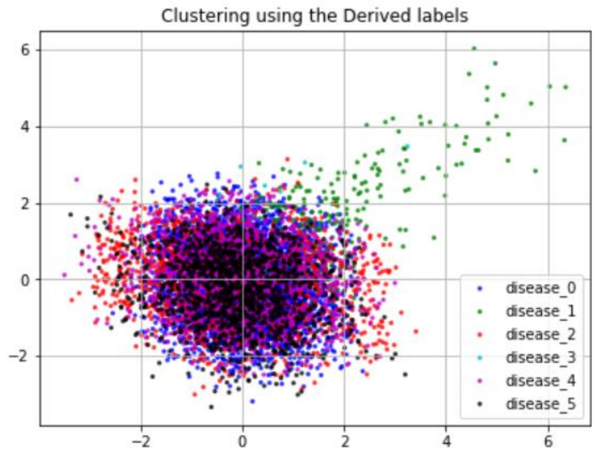
To visualize, we use the 2 PC (3d plot doesn't add much more information).

We used the whole training set of 9000 samples.

Using the TRUE samples (from the dataset):



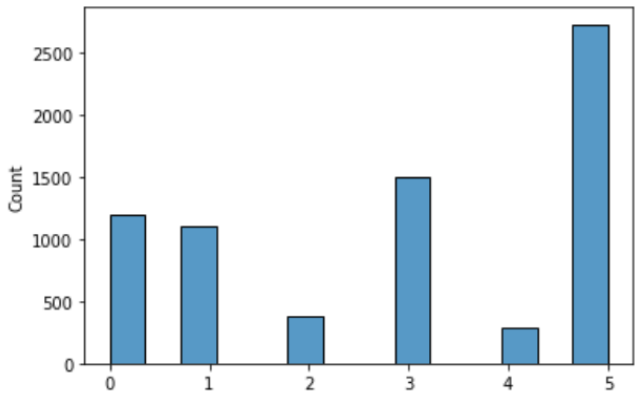
And using the Derived cluster indexes from the GMM:



We can see that GMM succeeded to discover some clusters (like the easy one which spans through big x and y values), but in general it doesn't correspond too much. We will later use this visualization to assert the precision of the classifier.

Class Imbalance

For the disease type, the distribution of the classes in original dataset is not balanced:



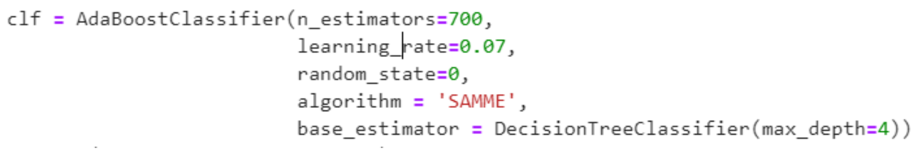
We handle the imbalanced data using SMOTE (Synthetic Minority Oversampling Technique), which created new samples for the minority classes, equating them with the majority class. The class imbalance handling was also added to the automatic training process.

Classification

We have implemented new gained knowledge to tackle the classification problem with more success than previously. We first concentrated on the Disease classification, since it was the most challenging task from the HW3.

1. Adaboost

We have used the AdaBoost classifier with various parameters, experimenting with the algorithm (sklearn offers SAMME, SAMME.R), number of estimators, base estimator and its parameters. The most successful parameters were the following:



```
clf = AdaBoostClassifier(n_estimators=700,
                         learning_rate=0.07,
                         random_state=0,
                         algorithm = 'SAMME',
                         base_estimator = DecisionTreeClassifier(max_depth=4))
```

Which gave around 86% on the test set for the Disease classification.

Pros

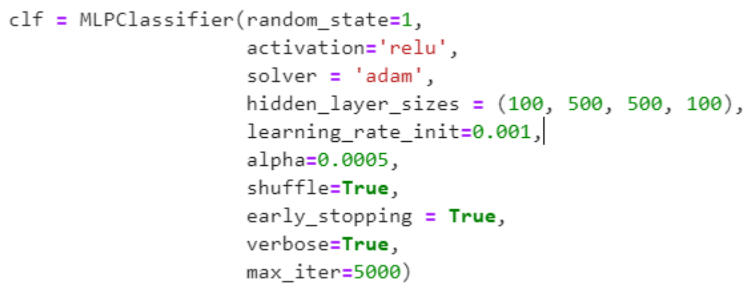
- Freedom of choosing base estimators.
- Easy to implement.

Cons

- Weak base estimators can lead to low margins overfitting
- Uniform noise can distort results

2. MLP

The sklearn library provides the MLP classifiers. We have as well examined different parameters variations; we found the following to be relatively fast converging and accurate:



```
clf = MLPClassifier(random_state=1,
                    activation='relu',
                    solver = 'adam',
                    hidden_layer_sizes = (100, 500, 500, 100),
                    learning_rate_init=0.001,
                    alpha=0.0005,
                    shuffle=True,
                    early_stopping = True,
                    verbose=True,
                    max_iter=5000)
```

In reality it would converge already after 50 epochs (iterations), with the accuracy of 89% on the test set, thus proving superior to the Adaboost.

Pros

- Flexibility of optimizing classification and regression problems.
- Free to choose any number of layers, thereby being able to solve complex pattern recognition.
- Improves with more data.

Cons

- No clear guideline on how to choose hyperparameters (layer sizes, amount of layers, learning rate, etc.)
- Computationally expensive.
- Requires large amounts of data to converge

3. MLP + Transfer Learning

In transfer learning, we used the following process:

a. First, train the binary classification on the Disease (0 = No disease, 1 = Any other disease). Use the SMOTE to fix class imbalance between classes (No disease, Disease)
b. Build a new model based on the trained model in (a.). Add new layers (only those layers will be trained)
c. Train the new model.

Using this process, we could achieve about 55% of accuracy on the Disease classification, which didn't improve our earlier tries.

Our analysis is that transfer learning did not improve results, because binary classification of disease/no disease did not generalize to differentiating between diseases, which led to low accuracy. This occurred even though the no disease class is largest, which is why we tried this method out in first place.

Transfer learning is usually used for NLP and Image Recognition and trained on much larger datasets. This is another reason for the low accuracy derived from the model.

(The code is available in transfer.py in the submitted work.)

4. Ensemble Learning

Ensemble learning is learning by a collection of estimators where we take a hard vote, regarding the best classification for each instance agreed on by most models, or a soft vote, summing over the probability assigned to each class and then picking the class with the maximum sum of probabilities assigned to it.

If the underlying collection of models is robust then ensemble learning provides a mechanism to increase certainty in the classification task.

Results

After running the automatic model selection, those are the models that were chosen by the process and their metrics on the test dataset.

Risk detection

The model chosen was an ensemble of four neural networks and AdaBoost with a decision tree base classifier with hard vote.

Chosen model : (AdaBoostClassifier(), {'n_estimators': [50, 100], 'base_estimator': [DecisionTreeClassifier(max_depth=1)]})

Metrics : f1_score

Value : 0.877 on validation set

F1 score is 0.77959 on test set.

Metric Selection

F1 score is ideal because we need a balance of low false negatives and high accuracy, because don’t want to tell healthy people they are at risk but also don’t want to make someone falsely certain he will have no consequences rom the virus

Spreader detection

Chosen model : (DecisionTreeClassifier(), {'max_depth': [5, 10, 15]}, True)

Metrics : recall_score

Value : 0.954 on valdation dataset

Recall on test dataset of 0.9466.

Metric Selection

Recall is the optimal metric to assess risk of spread, because a False Negative should be penalized harshly. A false negative can lead to massive spread so we would rather stay on the side of safety.

Disease detection

The model chosen was an ensemble of four neural networks and AdaBoost with a decision tree base classifier with hard vote.

```
Chosen model : (VotingClassifier(estimators=[('mlp1',
                    MLPClassifier(hidden_layer_sizes=(100, 150, 150,
                                                      100, 10),
                                  max_iter=150)),
                   ('mlp2',
                    MLPClassifier(hidden_layer_sizes=(50, 100, 200,
                                                      200, 100),
                                  max_iter=150)),
                   ('mlp3',
                    MLPClassifier(hidden_layer_sizes=(50, 100, 100,
                                                      200, 100),
                                  max_iter=150)),
                   ('ada',
                    AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=4),
                                       n_estimators=100))]), {}, True)
Metrics : accuracy_score
Value : 0.916 on validation set
```

Accuracy on test set: 0.69555

<u>Metric Selection</u>

Different diseases require varied interventions, therefore accuracy is vital.