Introduction to Machine Learning - 236756

# Homework #2

***Alexander Shender 328626114***

***Snir Hordan 205689581***

Technion - Israel Institute of Technology

## I. Part 2 - Dry Assignment

### A. Part 2a

The halfspace is homogeneous if the hyperpalce which defines them passed through the origin. Subsequently, the non-homogeneous halfspaces have to be defined by a hyperplance which is displaced by some parameter $b$ (bias) from the origin.

We include this bias $b$ term in the parameters $w$, thus bringing $w$ to $R^(d+1)$, which is denoted $z \in R^(d+1)$ in the assignment. But this term doesn't have to be multiplied by any input, thus we simply add '1' to the input vector $x$. The answer is thus:

$$g : R^d \to R^{d+1} = (x, 1)$$

$$z : R^d \to R^{d+1} = (w, b)$$

## B. Part 2b

First, some general notes on Datasets:

- the datasets A and C are not linearly separable in the original dimension
- the dataset B is linearly separable in original dimension

Some general notes on models:

- Perceptron can only classify linearly separable cases
- KNN can classify non-linear datasets, non-parametric
- Decision trees can classify non-linear datasets as well

YES = can fit NO = cannot fit

1. Perceptron
    - A: NO. The dataset is not linearly separable. This is actually similar to the XOR function, which cannot be learned by any linear model.
    - B:YES. The dataset is easily linearly separable.
    - C: NO. The dataset is not linearly separable. Only using the Kernel Trick (e.g. polynomial of 2nd order) can bring the dataset to a higher dimension, where they can be separated by a hyperplane.
2. KNN, $k = 1$
    - A: NO. As a simple example, it can be seen that on the far left on the border between the two groups there exists an orange dot really close to the blue ones. Using KNN with $k = 1$ it would classify this point as belonging to the blue group.
    - B: NO. The following point (orange) will be classified wrong, since its closest point is blue.
    - C: YES. From visual observation, all points would be classified correctly.
3. KNN, $k = 3$
    - A: NO. Same point as in the case with $k = 1$, this point would have a majority of 2 blue points in it's proximity, resulting in a wrong classification.
    - B: YES. From visual observation.
    - C: YES. From visual observation.

Figure 1: wrongly classified point at k=1

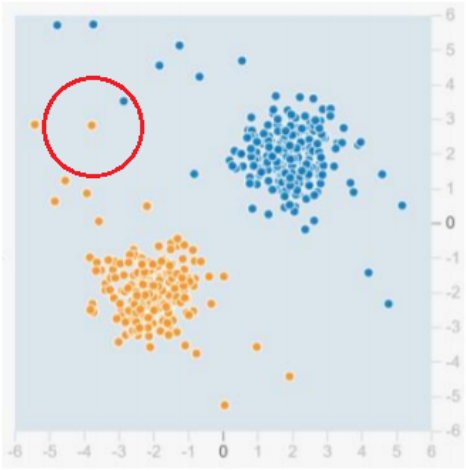


Figure 2: wrongly classified point at k=1

4. Decision tree

- A: YES. The decision tree can classify non-linear datasets, and in this example, it would require exactly 4 leaves to classify all the training dataset. It may look like this:

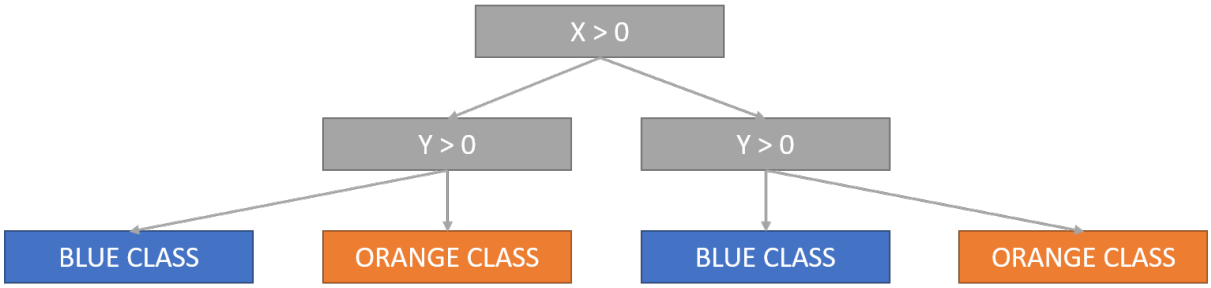



Figure 3: Decision tree for Dataset A with 4 leaves

- B: By having the limitation of 4 leaves, this dataset cannot be classified with 0 training error.
- C: Same as with B. By having limitless Decision tree depth this would be possible

## C. Part 3.1

To find the derivative with respect to the model parameters $w_j, j = 1...d$, we first write the cost function explicitly for both cases of the classification, since the cost function is defined diferently for the case where $y = 0$ and $y = 1$.

First, we denote shortly the sigmoid function with parameters $w$:

$$\sigma_w(x) = \frac{1}{1 - e^{-w^T x}}$$

Thus, we can rewrite the cost function:

$$C = cost(x_i, y_i, w) = y_i \cdot log(\sigma(x_i, w)) + (1 - y_i) \cdot log(1 - \sigma(x_i, w))$$

This way, the left side is active for the case with $y = 1$, and the right side for the case where $y = 0$. We also take $y_i$ out of the sigmoid function. It's worth noticing that the function is always negative except of the point where it is equal zero, when all predictions are perfectly correct. Thus, this is called the gradient ascend, since we are trying to "climb" to increase the "cost" function. Here the term "cost" has the opposite measning, i would better denote it "reward", since we are trying to maximize it.

Now we can calculate the derivatives with regards to the model parameters $w$. We summarize over all of the samples in a set.

Remembering:

$$\frac{\delta}{\delta w} log(f(w)) = \frac{1}{f(w)} (\frac{\delta}{\delta w} f(w))$$

$$\frac{\delta C}{\delta w_j} = \frac{1}{m} \cdot \sum_{i=1}^{m} ((y_i) \cdot \frac{1}{\sigma(x_i, w)} \cdot \frac{\delta \sigma(x_i, w)}{\delta w_{x_i}}) + ((1 - y_i) \cdot \frac{1}{(1 - \sigma(x_i, w))} \cdot \frac{\delta (1 - \sigma(x_i, w))}{\delta w_{x_i}})$$

We calculate the partial derivatives of the sigmoid function with respect to the model parameters using the chain rule and the hint in the assignment.

$$\frac{\delta \sigma(x_i, w)}{\delta w_{x_i}} = \frac{\delta \sigma(x_i, w)}{\delta (x_i, w)} \cdot \frac{\delta (x_i, w)}{\delta (w)} = \sigma(x_i, w) \cdot (1 - \sigma(x_i, w)) \cdot x_{i_j}$$

where $x_{i_j}$ is the j'th parameter of the sample $x_i$

inserting back into the equation:

$$\frac{\delta C}{\delta w_j} = \frac{1}{m} \cdot \sum_{i=1}^{m} ((y_i) \cdot \frac{\sigma(x_i, w) \cdot (1 - \sigma(x_i, w))}{\sigma(x_i, w)} \cdot x_{i_j} - ((1 - y_i) \cdot \frac{\sigma(x_i, w) \cdot (1 - \sigma(x_i, w))}{(1 - \sigma(x_i, w))} \cdot x_{i_j})$$

Shortening:

$$\frac{\delta C}{\delta w_j} = \frac{1}{m} \cdot \sum_{i=1}^{m} (y_i \cdot (1 - \sigma(x_i, w)) \cdot x_{i_j} - ((1 - y_i) \cdot \sigma(x_i, w) \cdot x_{i_j})$$

$$= -\frac{1}{m} \cdot \sum_{i=1}^{m} (y_i x_{i_j} - y_i x_{i_j} \sigma(x_i, w) - \sigma(x_i, w) x_{i_j} + y_i x_{i_j} \sigma(x_i, w) =$$

$$= -\frac{1}{m} \cdot \sum_{i=1}^{m} (y_i x_{i_j} - \sigma(x_i, w) x_{i_j})$$

$$\frac{\delta C}{\delta w_j} = \frac{1}{m} \cdot \sum_{i=1}^{m} (x_{i_j} \cdot (y_i - \sigma(x_i, w))$$

We are used to the matrix form. Converting the expression to matrix form:

$$\frac{\delta C}{\delta w} = \frac{1}{m} X \cdot (y - \sigma(X, w))$$

where $m$ is the number of samples in a set.

## D. Part 3.2

Since this is the gradient ascent algorithm, we do want to go in the direction of the steepest ascent, which is exactly the gradient. So, we update our parameters in the following way:

$$w^{t+1} = w^t + \eta \cdot \frac{1}{m} \cdot \sum_{i=1}^{m} (x_{i_j} \cdot (y_i - \sigma(x_i, w^t))$$

or in a matrix form:

$$w^{t+1} = w^t + \eta \cdot \frac{1}{m} X \cdot (y - \sigma(X, w^t))$$

where :

- $w^{t+1}$ - the parameters at the next iteration
- $w^t$ - the parameters are the current iteration
- $\eta$ - learning rate

## Code implementation

The supplied python code includes 4 files.

The main is “data_preparation.py”.

It loads the dataset, splits it into 6 datasets (train, validation, test) according to desired partitions set by user.

Then, some preliminary learning is done on the training set, e.g. the Principal Components from the PCA analysis. The number of principal components is, again, set by user. In our case, we chose 5.

Then, we created a custom Pipeline. It uses the basic Pipeline from the sklearn, but only applies the transformations (without ‘fit’ stage in the end). This way, it allows us to add stages to the Pipeline which modify each dataset. This Pipeline is applied on training, validation, and test set. Each stage in a pipeline makes some modification on some feature in the dataframe – it fills missing data, it changes some feature to categorical, or applies PCA transform, which drops the original features and leaves only the desired principal components. The final stage in the pipeline drops all the columns which have at least one NA value.

After the pipeline, each dataset is saved into a .csv file which is presented here.

# Feature Extraction Analysis

## Nominal Feature Visualization

**Feature Anomolies**

Some of the features have anomalies that skew the results greatly and therefore we truncated these features.

Looking at the bevaioral/physiological features:



**Social Media and Discipline**

Time on Social Media and Discipline Score have major outliers so those were truncated.

**Bmi Score**

The bmi score had data that was anomalous because it was physically impossible because bmi = weight/ (height)^2. Therefore, any bmi above 45 was truncated, and replaced the the mean value. The mean value was calculated on the training data, without the outliers.

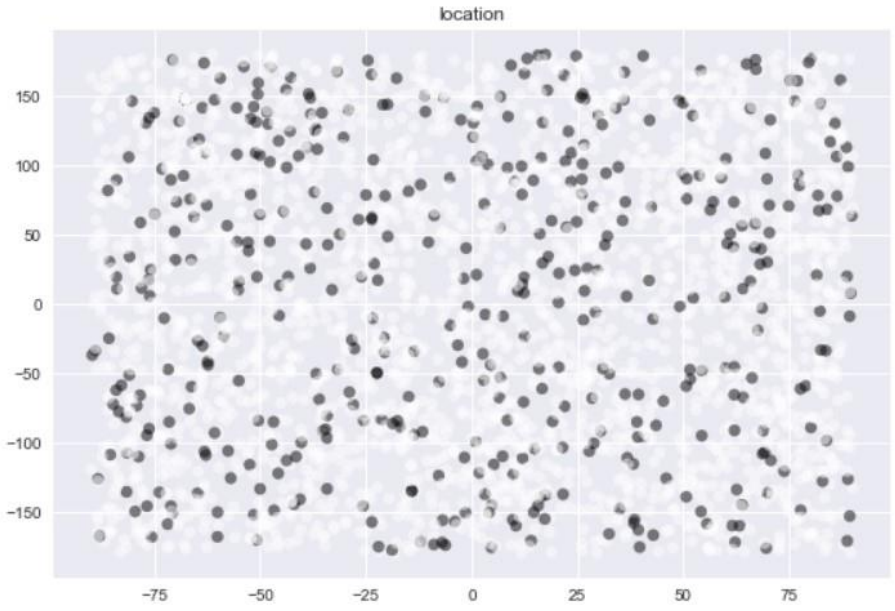
**Multi Label Classification**

We changed the classification to multi label with labels for risk, type of disease, and whether the person spreads the disease. This is important because some features like Syndrome Class indicate whether a person spreads or doesn't but isn't informative about the other labels. Those labels are also present in the final output csv files. Later the desired label for classification will be chosen.

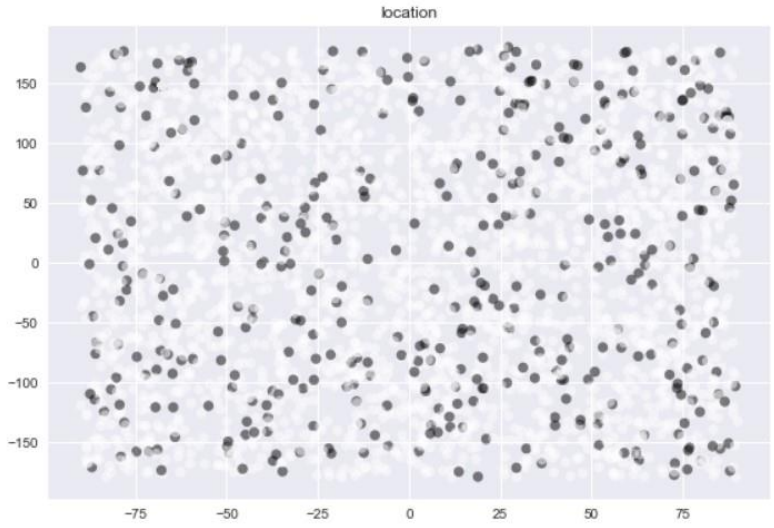
**Location data**

We observed the location to check whether it can give any reliable information about the type of disease. Maybe if people are congregated closely to each other then there is an outbreak. Yet observing the data, it can be seen there isn't a clear case of that:

Cases of Flue are in black and white not-flue:




Cases of covid in black and non-covid in white:



Graphs of other diseases are similar and show no specific area or zones of outbreak. Therefore the location feature is redundant.

Also, the likelihood of spread or risk are not related to the location of the person, but rather to physiological, behavioral traits.

**Scaling**

For scaling we tried to use Quantile Transformer from sklearn but because it truncated too much of the data the results were flawed. Eventually we used standard scaling of (x- (avg))/(std. deviation) on the pcrResults in the PCA.

**Expert Knowledge**

From general knowledge, the address, patient ID, date of test, number of cousins are redundant as well. Also, the professions have too many types in the dataset and from human knowledge are irrelevant.

Plotting SyndromeClass vs. the TestResultsCode we observed the following:

Where the right-hand side is the TestResultsCode and the top side are the syndrome classes. We saw that non-spreading people mostly didn't exhibit syndromes 1 and 4. Then we chose to keep the feature.

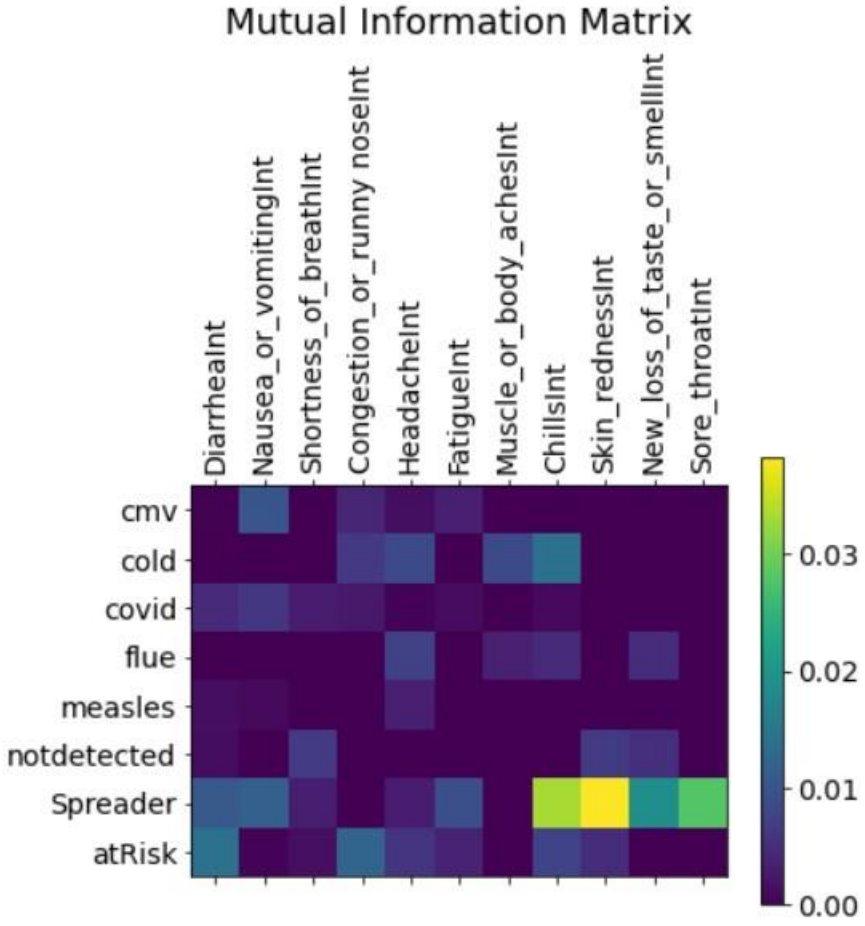
We chose to remove AvgHouseholdExpenseOnPresents, `AvgHouseholdExpenseOnSocialGames, AvgHouseholdExpenseParkingTicketsPerYear, AvgMinSportsPerDay, AvgTimeOnSocialMedia and AvgTimeOnStuding.`

By Occam's razor principle the learning will be more efficient excluding them, because the simpler model that takes into account only the most relevant features only will likely be better performing. The above features are immaterial to communicable diseases, because they focus on the household and activities that don't pose much risk to catching diseases.

Also `AvgTimeOnSocialMedia and AvgTimeOnStuding,` have too few samples of data. `AvgTimeOnSocialMedia` entries that are not Nan and AvgTimeOnStuding has 750 entries that aren't NaN. (in the entire dataset). This shows they aren't sufficiently informative to be taken into consideration.

**Mutual Information**

We used Mutual Information to assess SelfDeclarationOfIllnessForm and the occurrence of diseases. We found very low mutual information correlation between a syndrome reported and one of the diseases in the test results, but there was some correlation if the person was spreading. as seen :


Mutual Information Matrix
DiarrheaInt
Nausea_or_vomitingInt
Shortness_of_breathInt
Congestion_or_runny noseInt
HeadacheInt
FatigueInt
Muscle_or_body_achesInt
ChillsInt
Skin_rednessInt
New_loss_of_taste_or_smellInt
Sore_throatInt
cmv
cold
covid
flue
measles
notdetected
Spreader
atRisk
0.03
0.02
0.01
0.00

# PCA transform

We use the PCA on the “pcr results” measurements. Since there are 16 such measurements, we want to find the most dominant Principal Components.

First we full the missing data with the median of each “pcr result” caregory, since they have gaussian-like distribution.

Then, we use the Scaler function to normalize the values of all the measurements.

Then, we define the number of Principal Components we want to save. We choose 5 arbitrarily, later this number will be adjusted according to the classifier performance.

We project the feature vectors of “PCR results” onto the 5 Principal Components. We get 5 new features, which are in fact linear transformation of the original features onto the 5 Principal Components.

So we do want to **save all 16 pcr results features,** but those will be transformed into only 5 new features, which are the values on the 5 biggest Principal Components.

## Sequential Forward Selection

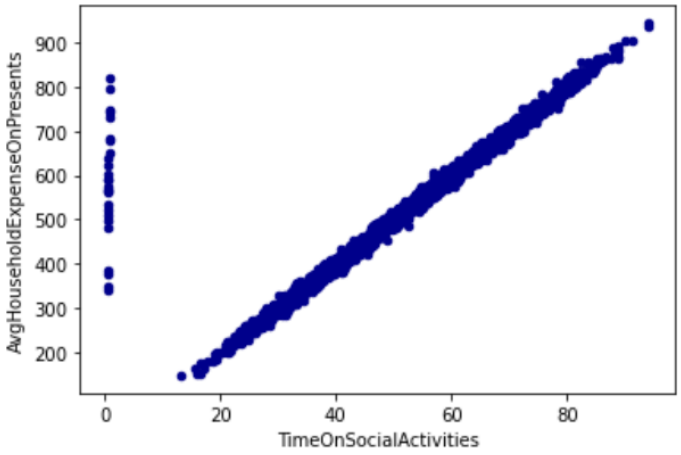
In order to determine which remaining features were of importance, we chose to use SFS. We chose the top three from considerations of Occam's Razor, i.e. the simpler model is usually the best one.

We used the K-Nearest Neighbors algorithm as our scoring function, because it requires no assumptions on the data, and only one hyperparameter, and there are many distance criteria to choose from. We chose to use the scoring function with 2 neighbors in order for it to generalize but not to be too computationally expensive.

Uwe found the three most important features other than previously mentioned were BMI, Time On Social Activities and Discipline Score, which make sense in terms of expert knowledge. Those features all have missing values, which have to be replenished. The BMI feature refill was explained before.

**Time on Social activities**

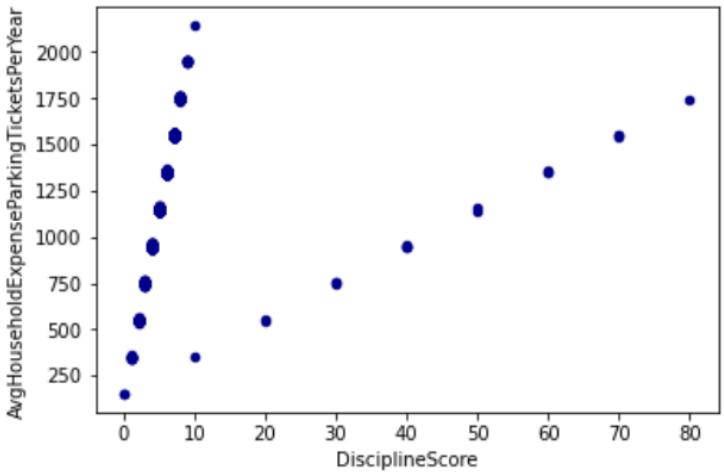
A very good linear dependency was found between the Time on social activities and The Expenses on the presents. First, it lets us refill the missing values. And shows how one of the 2 features is actually redundant



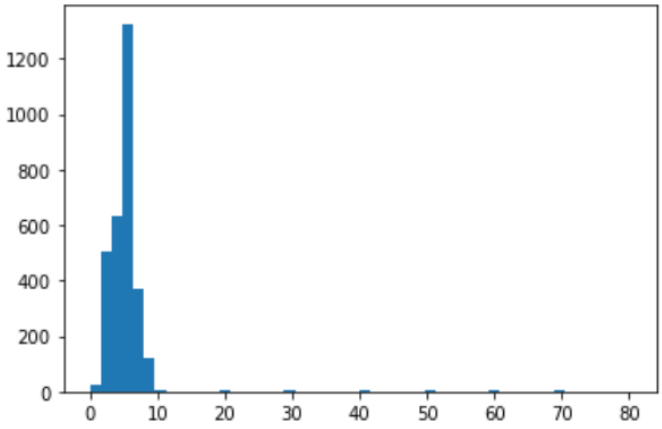

So if the data for Time on Social activities was missing, it was replenished from the Presents expenses. If the data was missing there as well, it was filled with a mean value calculated from the training set.

**Discipline Score**

Another linear dependency was found, between discipline score and expenses on parking tickets:




First, we can see 2 lines, which is strange, so we also take a look at Discipline score histogram:




We can see that presumably the values above 10 are outliers, and from the first chart we can assume that there is redundant 0 (zero) in the number. E.g. 80 instead of 8, etc.

So, any value above 10 was divided by 10. Any missing value was refilled with the ParkingExpenses. If value was still missing, was refilled with the mean calculated from the training set

## Imputation

We tried using imputation using linear regression to fill up nan values. Then using r2 score, we checked if filling up using LR was better than just filling up with the mean.

We chose 'AvgMinSportsPerDay', 'DisciplineScore', 'HappinessScore' , yet we didn’t get meaningful r2 scores.

## Pros and Cons – Sequential Forward Selection

**Pros**

- **Simple to implement and intuitive**
- **Can specify in advance the number of important features desired**
- **Can choose from variety of scoring funtions**

**Cons**

- **Difficult to obtain same feature set from slightly modified data. Reproducibility is hard to achieve.**

## Pros and Cons k-NN

**Pros**

- **Many distance criteria**
- **No assumptions on data**

**Cons**

- **Class imbalance can have outsized effect**
- **No clear guideline on choosing number of neighbors**

## Conclusion

The features that were decided to be left:

1. BMI, DisciplineScore, TimeOnSocialActivities – from the SRS.
2. All the PCR results (16 features). We do not count them as 16 features, since later after the PCA projection, they transform into 5 features. (number of chosen Principal components)
3. Syndrome Class
4. Sex, BloodType – those were chosen arbitrarily, from the common sense, since perhaps they can have a good impact on the classification
5. SelfDeclarationOfIllnessForm – as stated before, some of those have a good MI scores with the labels, so we have to consider them on course

So we have 3 + 5 (16) +1 + 2 + 1 = 12 chosen features

Pay attention, that the CSV files have already the SelfDeclarationOfIllnessForm divided into various different one-hot declarations. And the result labels as well.