The predictability of life course

Applied Statistics Project

Students
MASSIN Keryann
VEILLON Juliette
ANDRU Kilian
LACOUR Xavier

Supervisors
M. TROPF Felix
M. PETEV Ivaylo

Introduction

I – Presentation and Data cleaning

II – Creation of the Health Index

III – First machine learning procedures

IV – Second machine learning procedures

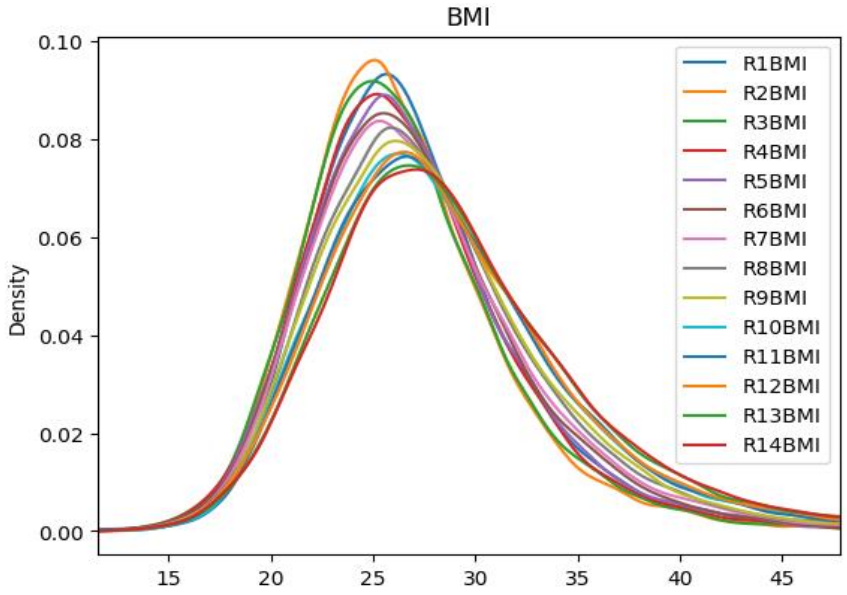
Conclusion

I – Presentation and Data cleaning

- 3 databases from the *Health and Retirement Study*
  - Socioeconomic data
  - Genetic data
- 42,233 individuals and 15,104 variables
  - Need for dimension reduction
  - Imputable and non-imputable missing values
- Reduction of the number of variables to 4,147

II – Creation of the Health Index

- Global Health Index to summarize health-related information
- Selection of 27 pertinent variables
  - Descriptive statistics



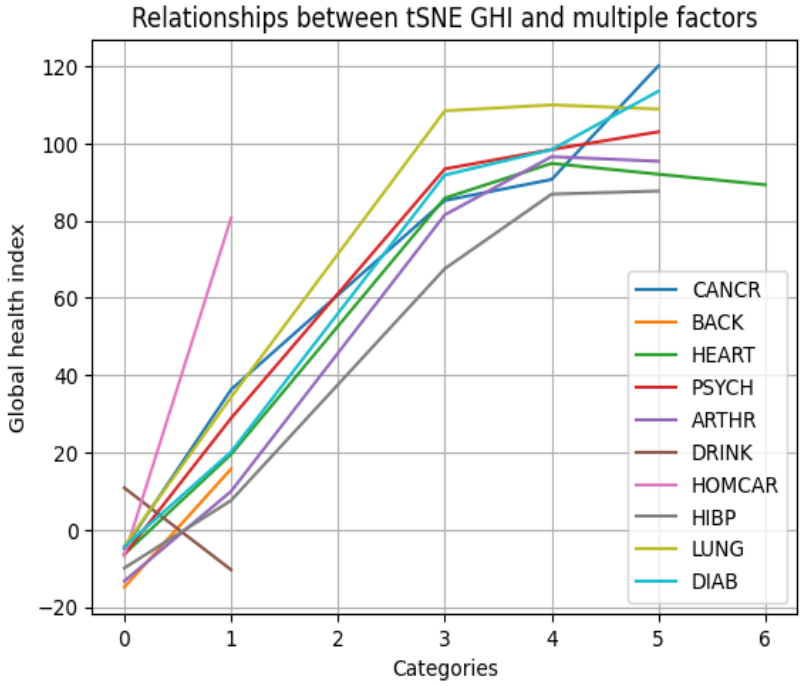
Density of the variable BMI for each wave

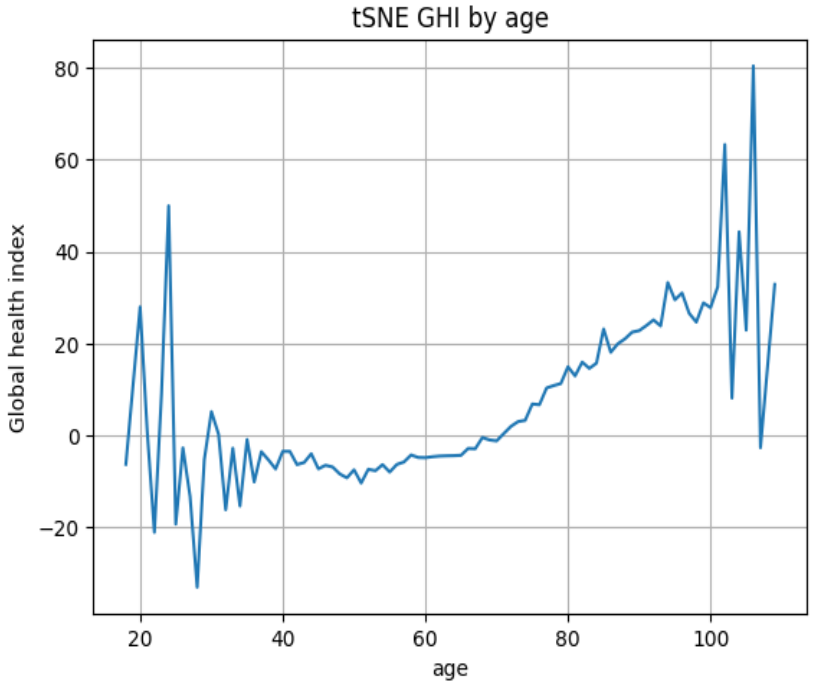
	R1HEART = 0	**R1HEART = 1**
R1HIBP = 0	7094	690
R1HIBP = 1	3892	976

Contingency table of heart issues against hypertension in wave 1

II – Creation of the Health Index

- T-distributed Stochastic Neighbor Embedding
- Tests of robustness





III – First machine learning procedures

- $n \approx p \Rightarrow$ Standard methods would not converge
- **Idea:** Select $p' \ll p$ predictors beforehand
  - How? Lasso
  - Problem: a lot of missing data
- **Solution:** modified version of the Lasso
- **State-of-the-art:** CoCoLasso vs HMLasso
  - HMLasso better for our purpose
  - But not implemented in Python…
- **So we implemented it!**

III – First machine learning procedures

- HMLasso in Python
  - CVXPY library → solver
  - Scikit-learn like interface
- **Useful** to select predictors
- **On our data, huge gain**
  - $p = 4,147 \Rightarrow p' = 1500$

```python
# Data
X, y = get_Xy(n=10000, p=10, replace_rate=0.4) # replace_rate = 40% of missing values

# Scaling the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = y - y.mean()

# Fitting the HMLasso
lasso = HMLasso(mu=1, alpha=1)
lasso.fit(X_scaled, y_scaled)

# Estimator
print(lasso.beta_opt)
```

```
[ 2.025e+02 -6.120e+01  1.430e+02  5.478e+02  1.707e+02 -0.000e+00
  2.000e-01 -2.000e-01  0.000e+00 -1.400e+00]
```

We can drop columns 6 and 9

IV – Second machine learning procedures

- **Two kind of methods**
- **Linear methods:**
  - 2SLS
  - Within regression
- **Tree Based Methods:**
  - XGBoost
  - RandomForest

IV – Second machine learning procedures
Linear methods – 2SLS

- **Principle**
  - Create a different lasso regression for each wave
  - We use the previous estimated GHIs to help determine the next one

$$\widehat{GHI_i} = \widehat{\beta}_i X_i + 1_{[\![2,14]\!]}(i) \sum_{j=1}^{i-1} \delta_j \, \widehat{GHI_j} + 1_{\{14\}}(i)\hat{\gamma} G$$

- **How to modulate the model?**
  - Penalisation
  - Lasso-type used
  - Type of imputation

IV – Second machine learning procedures
Linear methods – 2SLS

- **Results:**

Data	Testing set's R^2	Testing set's RMSE
With genetic data	0.284062	48.032847
Without genetic data	0.276727	48.278280

Results of the multiple 2SLS lasso regression
($\mu = 0{,}5$ and mean-imputation)

→ There is no real improvement of the R^2

IV – Second machine learning procedures
Linear methods – Within Regression

- **Linear regression on Panel data :** $\forall i,\ \forall t :\ Y_{it} = \beta_0 X_{it} + \alpha_i + \varepsilon_{it}$
  - α_i : Fixed effect for each individual, $E(X_{it}\alpha_i) = 0$
  - ε_{it}: Error term ≡ Shocks on General Health Index (Y_{it})
- **Fixed effect estimates**
  - Strict exogeneity : $\forall(t, t')$, $E(X_{it}\varepsilon_{it'}) = 0$
  - Within regression ($U_{it} - \frac{1}{T}\sum_{t=1}^{T} U_{it}$) or First difference regression (U_{it} - U_{it-1})

IV – Second machine learning procedures
Linear methods – Within Regression

- **Machine learning method**
  - Stacked lasso selection :

$$X_1 \xrightarrow{Lasso\ (GHI_1)} X'_1\ ,\ \ (X'_1, X_2) \xrightarrow{Lasso\ (GHI_2)} X'_2\ ,\ \dots\ ,\ (X'_{13}, X_{14}, G) \xrightarrow{Lasso\ (GHI_{14})} X'_{14}$$

  - Within regression to predict GHI_{14} with X'_{14}
- **Parameter optimization**
  - Only parameter : Lasso penalization μ
  - Optimization with a validation set : light over-fitting

IV – Second machine learning procedures
Linear methods – Within Regression

- **Results :**

Data	Validation set's R^2	Testing set's R^2
With genetic data	0.0832	0.0528
Without genetic data	0.0830	0.0486

First method (Within transformation after Lasso selection)

- **Genetics variables :**
  - 3 genetic variables kept in this method

IV – Second machine learning procedures
Tree Based methods – XGBoost

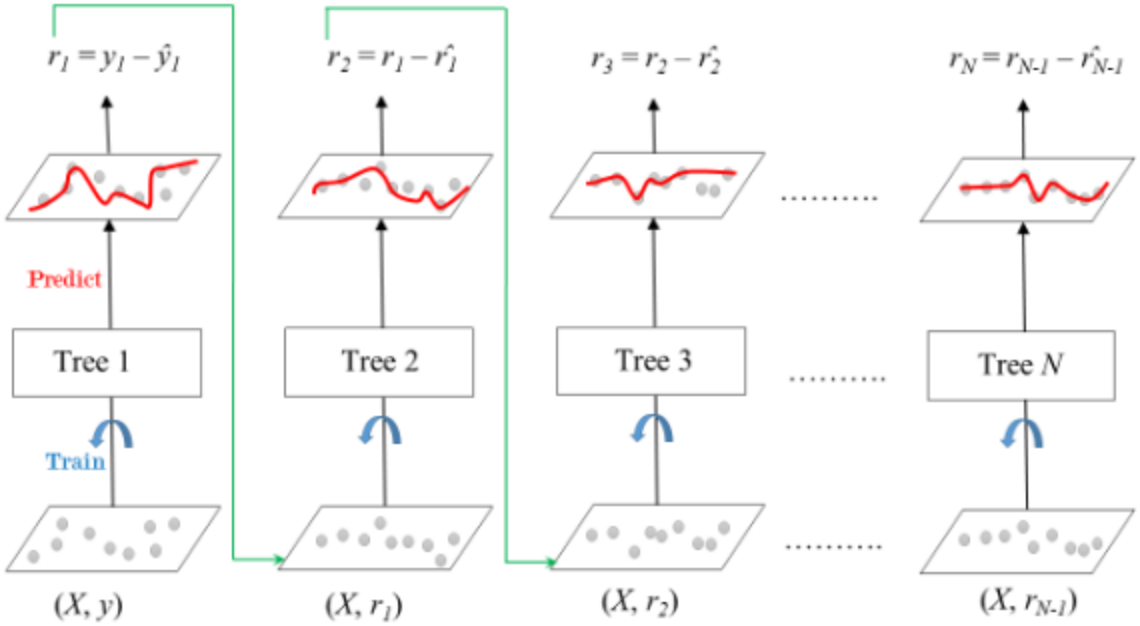
- **Fragile Family Challenge:** XGBoost provided powerful results
  - Let's give it a try!
- **What is it?**
  - Gradient Boosting Method

IV – Second machine learning procedures
Tree Based methods – XGBoost

- **Fragile Family Challenge:** XGBoost provided powerful results
  - Let's give it a try!
- **What is it?**
  - Gradient Boosting Method
  - **Idea**: greedily construct

$$f(x) = \sum_{i \leq N} Tree_i(x)$$

  - **XGBoost:** algorithm of Gradient Boosting that handle missing values
  - **Run extremely fastly!**



Source: https://www.geeksforgeeks.org/ml-gradient-boosting/

IV – Second machine learning procedures
Tree Based methods – XGBoost

- **On our data**
  - Regression
  - Optimized hyperparameters using cross-validation
  - **Results:**

Good results!

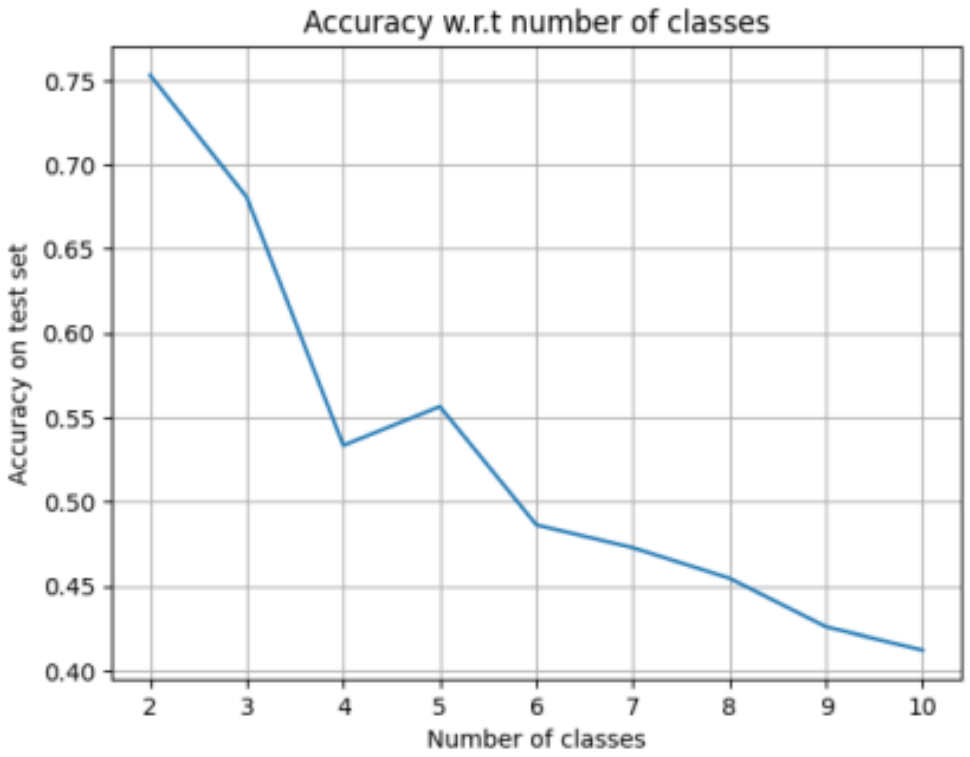
Data	mean R^2	std R^2	mean RMSE	std RMSE
all data	0.3688	0.0285	41.87	0.97
only previous ghi	0.3387	0.0267	42.74	1.13
only socioeconomic data	0.2130	0.0252	46.71	0.95
only socioeconomic and genetic data	0.2118	0.0250	46.85	1.09
only socioeconomic data and previous ghi	0.3672	0.0279	42.00	1.16

Table 5: XGBoost regressor - results over 100 simulations

Unable to identify the effect of genetic variables

IV – Second machine learning procedures
Tree Based methods – XGBoost

- **On our data**
  - Regression ⇒ **what about classification?**
  - Is GHI = 99 really worse than GHI = 100?
- **For 2 categories** *{bad health, good health}*,
  - 75,31% of accuracy
  - 1,51 times better than dummy classifier
- **For 10 categories**,
  - 41,21% of accuracy
  - 4,12 times better than dummy classifier!



IV – Second machine learning procedures

Tree Based methods – Random Forest

- **On our data**
  - Regression
  - Optimized hyperparameters using cross-validation
  - **Results:**

Strong prediction power!

Data	mean R^2	std R^2	mean RMSE	std RMSE
all data	0.3970	0.0029	40.85	0.01
only previous ghi	0.3767	0.0209	41.86	0.38
only socioeconomic data	0.1608	0.0325	47.68	1.46
only socioeconomic and genetic data	0.1660	0.0203	46.42	0.44
only socioeconomic data and previous ghi	0.3608	0.0245	41.87	0.10

Table 7: Random Forest regressor - results over 2 simulations

IV – Second machine learning procedures

Tree Based methods – Random Forest

- **On our data**
  - Classification
  - **Results:**

Random Forest < XGBoost

Number of classes	Random Forest	XGBoost	Dummy classifier	Forest/Dummy ratio
2	0.7515	0.7531	0.5	1.46
3	0.6044	0.6807	0.3333	1.71
4	0.4926	0.5334	0.25	1.83
5	0.4294	0.5565	0.2	2.03
6	0.3897	0.4862	0.1667	2.11
7	0.3618	0.4726	0.1429	2.38
8	0.3559	0.4546	0.125	2.64
9	0.3235	0.4257	0.1111	2.66
10	0.3015	0.4121	0.1	2.67

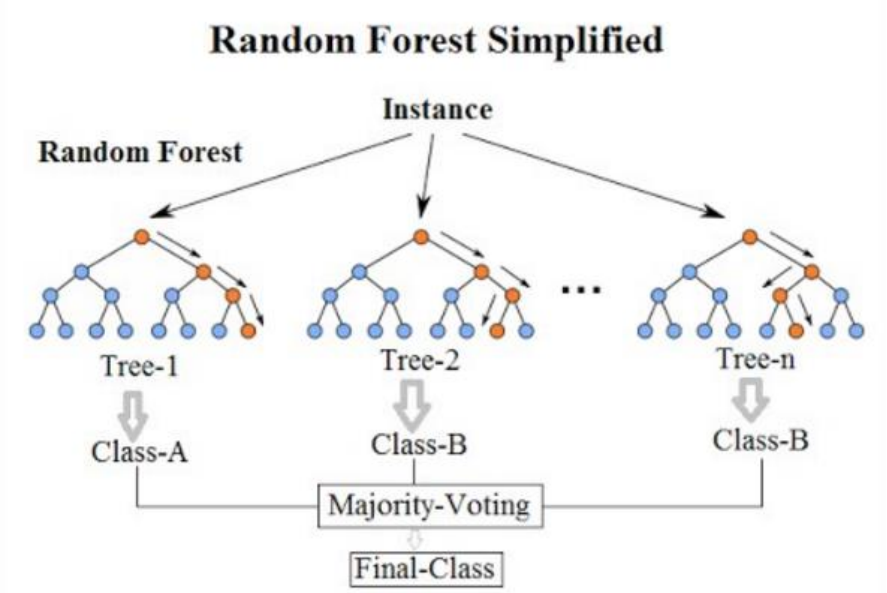
Table 8: Random Forest classifier - results over 10 simulations

Conclusion

- **Creation of an index to summarize health**
- **Application of linear methods and trees based methods**
- **Success of prediction** (XGBoost classifier and Random forest regressor)
- **Usefulness of genetic variables ?**
- **Possible improvements**

Annexe
Tree Based methods – Random Forest

- Boosting works. **What about bagging?**
  - **Random Forest** vs Gradient Boosting
- **What is it?**
  - **Idea:** parallelise the training of decision trees
  - Wisdom of crowds ⇒ Majority-voting to predict a class
  - **How?** Database boostrap



Source: https://en.wikipedia.org/wiki/Random_forest