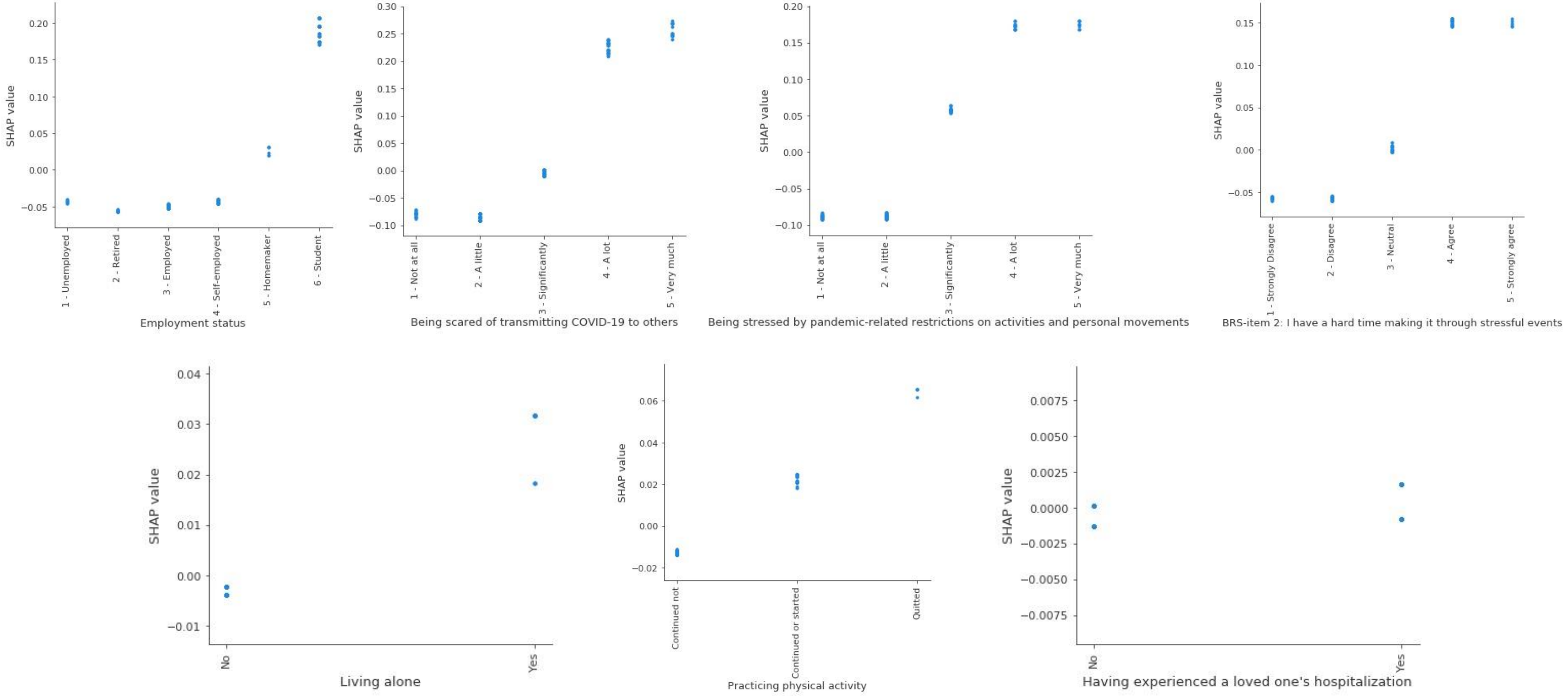
Predicting new-onset psychiatric disorders throughout the COVID-19 pandemic: A machine learning approach

Supplementary Material

Figure S1: Variables included in the final ML predictive model and average of the absolute SHAP values for each variable, ordered by their relevance to the model (test dataset, second wave)



Levels of the variables plotted against the associated SHAP values in the second wave: visual representation of the relationship between each variable and the risk of having at least one provisional psychiatric disorder (PPsyD) that has been modeled by the algorithm.

ML: machine learning; SHAP: SHapley Additive exPlanations technique

Table S1: Geographical distribution of the participants included in the study

	First wave (N = 500)		**Second wave (N = 236)**	
Distribution§	N	%	N	%
North-western Italy	348	69,6	153	64,8
North-eastern Italy	45	9,0	29	12,3
Central Italy	47	9,4	18	7,6
Southern Italy	47	9,4	25	10,6
Italian islands	13	2,6	11	4,7

§ variable not included as potential predictor, because the majority of participants were from north-western Italy; N: number

Machine learning methodology

Variables

For variables that were potentially useful as predictors in the model, we included all the individual information concerning aspects preceding the pandemic or representing its direct consequences, such as pandemic-related personal experiences.

We decided *a priori* to remove variables with greater than 20% missing values in the training set (i.e., data from the first wave survey), whereas all variables with missing values equal to or less than 20% remained included in the analyses.

All the 46 variables suitable to be included as potential predictors during the training of the algorithm are reported in Table 1 (section Results) and in this online supplement (Tables S2-3). The categorical variables were re-coded with the label-encoding strategy, namely, all cases of each categorical variable were assigned an integer. If the variable was ordinal, the class-to-integer conversion respected the order of the classes.

Some questions were not administered to all participants because they were not relevant (e.g., questions regarding relationship with children were not administered to participants without offspring). In this case, the value was coded 0 (“not relevant”) and the other classes of answers started from value 1. This coding strategy for categorical variables is justified by the use of a tree-based ML technique.

The missing values were imputed using the MissForest technique {1}, and implemented with the *IterativeImputer* function of the Scikit-Learn library version 0.22.2 {2}, using Random Forest {3} as an estimator. The imputation model was first developed using the training dataset and was then applied to the test dataset.

Gradient boosting technique

Boosting is an ML technique that produces a prediction model in the form of an ensemble of several consecutively developed prediction models. In the current study, we used decision tree models, which are commonly used within the gradient boosting ensemble technique. Several decision trees are iteratively built, each consecutively trained to reduce the misclassification of the previous decision trees {4}. The final prediction is the result of a weighted sum of the prediction performed by all the consecutive decision tree models, which can be as many as hundreds.

The current study used the implementation of gradient boosted decision trees (GBDT), provided in the eXtreme Gradient Boosting (XGBoost) library {5}.

Hyper-parameter optimization

Several hyper-parameters need to be defined for the GBDT model. Different values of these hyper-parameters lead to different predictive performances by tuning the training process. The aim is to identify the configuration that produces the best possible performance when applied to cases that are not part of the training set. To optimize such hyper-parameters, the algorithm was first trained with 40 random hyper-parameter configurations. Subsequently, 60 further configurations were progressively estimated with a Bayesian optimization approach, which estimates the hyper-parameter configuration that will maximize the performance of the algorithm. This estimation was performed with Gaussian Processes, as implemented in the Scikit-Optimized library (https://scikit-optimize.github.io/).

The Area Under the Receiving Operating Curve (AUROC) was used as the performance metric to be maximized during the hyper-parameter optimization process. The AUROC value is 0.5 when the algorithm makes random predictions, and 1 in case it is always correct in making predictions. AUROC is not directly affected by class imbalance.

Cross-validation

The aim is to develop an algorithm that achieves the best possible generalized performance and that also performs well beyond the cases used in the training process. Cross-validation provides an estimate of such generalized performance for every hyper-parameter configuration. In cross-validation, the training sample is divided into several folds of cases that are separated from the training process, with training iteratively performed with the remaining cases. After the training, the algorithm is finally applied to the separated cases.

In this study, we applied a 10-fold cross-validation procedure, stratifying (i.e., balancing) for the percentage of at least one new-onset provisional PsyD (PPsyD) in each fold. Finally, the ten performance estimates of the algorithm available for each hyper-parameter configuration were averaged to provide a final point estimate of the generalized performance. The hyper-parameter configuration that demonstrated the best average cross-validated AUROC was retained and was used to re-train a single algorithm with the entire training sample.

Selection of variables

In order to select an optimal subset of the initial 46 variables to be used as predictors in the ML algorithm, we used the Minimum Redundancy, Maximum Relevance (mRMR) approach {6}. This technique provides a ranking for the progressive inclusion of the available variables, indicating for each step (in the current study, from using a single variable to all 46 variables) which variables should be additionally included. The mRMR approach identifies which variable to include next by simultaneously taking into consideration the association with the output for each of the remaining variables (maximum relevance, i.e., the variable whose addition would bring maximal addition of predictive information) and the association with variables that have been selected at previous iterations (minimum redundancy, i.e., the variable whose addition would bring minimal redundancy to the predictive information already provided by the variable that has already been included).

In this study, the above-mentioned hyper-parameter optimization and cross-validation procedure was performed 46 times, each time considering as predictors a subset of size 1 to 46 variables (all variables) as indicated by the mRMR procedure. For each feature subset, the result of each hyper-parameter search is a final algorithm based on the optimized hyper-parameter configuration, with an associated average cross-validated AUROC. The subset that resulted in a better average cross-validated AUROC was chosen as the optimized set of variables to use, and the associated algorithm was retained as the final algorithm.

Reference

1. Stekhoven DJ, Bühlmann P: MissForest—non-parametric missing value imputation for mixed-type data. Bioinformatics 2012; 28:112–118
2. Pedregosa F, Michel V, Grisel O, et al.: Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot, 2011
3. Breiman L: Random Forests. Mach. Learn. 2001; 45:5–32
4. Friedman JH: Greedy function approximation: A gradient boosting machine. Ann. Stat. 2001; 29:1189–1232
5. Cheng B, Zhang D, Chen S, et al.: Semi-supervised multimodal relevance vector regression improves cognitive performance estimation from imaging and biological biomarkers. Neuroinformatics 2013; 11:339–353
6. Hanchuan Peng, Fuhui Long, Ding C: Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans. Pattern Anal. Mach. Intell. 2005; 27:1226–1238