Project's report
Scientific computing and object oriented programming

Alessandro Reami

August 17, 2023

0 Preface

The dataset that we will use isn't already public, it's been collected by: prof. Silvia Ferrari, doc. Rosaria Di Lorenzo, doc. Diego Dragone and doc. Matteo Reami. This dataset it's been already used for M. Reami's master thesis (advisors: S.Ferrari and R. Di Lorenzo), it was about a standard statistical analysis with Shapiro-Wilk test, Skewness-Kurtosis test, Shapiro-Francia test, Kruskal-Wallis, mean and standard deviation. On that thesis it's not been used any of the machine learning tool that we will use for this new project but we will recall some of their results about the statistical significance of some feature. The reader can find the thesis In the *Dataset* directory of this project.

1 Project's goal: predict Revolving Door hospitalizations for Modena's SPDC

In Modena's SPDC ("Servizio Psichiatrico di Diagnosi e Cura", Service of Psychiatric Diagnosis and Care), as in many other SPDCs, it's possible to notice the presence of some patients who tend to have several hospitalizations in short time periods [2][5][6], usually psychiatrists refere to this behaviour using the term *Revolving Door* but without a standard definition. We will choose for this project the same one adopted for the dataset generated by Modena's SPDC's psychiatrists [3].

This dataset collects the discharge letters written at the SPDC between January 1^{st}, 2017 and December 31^{th}, 2022, which means that it collects almost all the letters for a period that last six years, just few of them are been removed by the SPDC's doctors due to the huge presence of missing data.

In this project we will test if it's possible to predict the revolving door behaviour of a patient just from its last discharge letter, consider the 0-1 error that this prediction will produce on a test set and the associated confusion matrix. It's easy to understand that provide a model able to predict, just from the discharge letter, if a patient will be a revolving door is a powerful tool for a psychiatrist and it may lead to a more appropriate outpatient follow-up. This information may easily lead to a better quality life for the patients: the possibility to reduce TSO and TSV treatment (mandatory medical treatment

and voluntary medical treatment) while increasing a more specific local medicine follow-up with a more suitable therapy.

To achive this goal we will test how is predictable the patients' revolving door behaviours using a random forest model, we will in particular use the C# programming language and the Accord libraries dedicated to the machine learning.

2 Dataset: a brief description

The dataset, not yet in the public domain, is been assembled by the Modena's SPDC with a single-centre retrospective analysis and it collects the discharge letters writted during six years. We recall here that this SPCD is placed at the OCSAE (civil hospital "Sant'Agostino d'Estense") of Baggiovara (Modena) and that the letters are collected using its SIO (hospital informatic system). Initially it was collected to understand how the Covid disease and the following lockdown modified the behaviour of patients and doctors: in which cases it increased the number of needed treatment or it got worse the psychiatric condition of the people. Now we want instead check if it's possible to use the same dataset to study some other important aspects as: revolving door behaviour, patients' suicide attempts trend and effectiveness of the TSO treatment ("Trattamento Sanitario Obbligatorio"- mandatory medical treatment). For this paper we will only focus on the first topic.

The dataset collects 2954 discharge letters from 1903 different patients (on average 1.55 letters for patient) that are almost the total number produced between January 1^{st}, 2017 and December 31^{th}, 2022 (just a few number of them are not included in the dataset due to the huge presence of some missing data that were impossible to recover). A preliminary pre-process of the data is been made by the same SPDC, recovering, when it was possible, the missing data: some of them are been recovered from another letter of the same patient or asked directly to a doctor for the remembered informations. Anyway some data are still missing and we will have to remove some letters from the dataset (more detail at Section 5) ending up with 2928 letters that is the 99.12% of the dataset.

The letter cointained in the dataset are standardized and have the following fields (in boldcase the ones that we will use, see why in following section):

1) **hospitalization counter**;
2) **gender**;
3) **age**;
4) year of discharge;
5) season of discharge;
6) **working condition**;
7) presence of a support administrator;
8) **housing condition**;
9) nationality;
10) **schooling**;
11) admitting motivation;
12) **double diagnosis** (notice that this term is refered **only** to addiction: drugs and gambling);
13) **type of treatment** (TSO or TSV);
14) previous charge;
15) **discharge diagnosis**;

16) **use of NL, LAI, AD, BDZ, mood stabilizers or other drugs** (NeuroLeptics, Long Active Injectable Antipsychotics, AntiDepressants, BenzoDiaZepines);

17) mono or polytherapy;

18) **discharge destination**;

19) comorbidity;

20) covid positivity during the hospitalization;

21) **days of hospitalization**;

22) days of TSO;

23) aggressive behaviours during the hospitalization;

Moreover, during the collecion of the data, it's been computed the field **Revolving Door behaviour**, that we will use as label for our machine learning algorithm, containing 1 if the patient will be defined as revolving door in future and 0 if it will not. For this dataset it's been defined as revolving door every patients with three or more hospitalizations in less than 365 days, this definition lead us to have 2954 discharge letters and 389 (13.18%) of them presenting a revolving door behaviour, Figure (1).

It's possible to find a complete legend for the dataset (partially translated in English) at the last sheet of the *Dataset.xlsx* file in the *Dataset* directory of this project. Some medical aspect of the dataset are here neglected but it's possible to have a deeper knoledge of the dataset reading it.

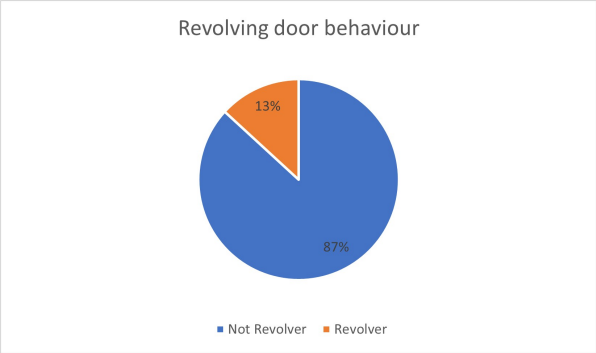

Figure 1: Presence of revolving door behaviour in the letters.

3 Machine Learning: Classification with random forest and 0-1 loss function

In this project we will use a binary random forest classifier with 0-1 loss function already implemented in the Accord.Net framework, this is a supervised machine learning algorithm designed for binary classification tasks, where the goal is to predict one of two possible value of the label for each input data point. We chose this model since it offers several advantages, including robustness against overfitting, good generalization, and the possibility of being parallelizable [1][8].

In this case we will use the discharge letters as input data point and the label to predict will be the Revolving door field. This algorithm utilizes the concept

of a "forest" of decision trees to make predictions while minimizing the 0-1 loss function.

Firstly we notice that the label is enough common in the dataset: the 13% of the input data points present a revolving door behaviour, so we randomly partition the whole dataset in training data and test data following a ratio of 85%.

Then we have to tune the hyperparameter of the model: the forest is composed of multiple decision trees, decided by the `NumbTree` variable, each one trained on a subset containing a percentage of the training data (`Ratio`) and using a portion of the features (`Coverage`), this randomness is wanted since it's useful to reduce overfitting and increasing the diversity of the individual trees. In order to choose the best value of the hyperparameters `NumbTree`, `Ratio` and `NumbTree` we will perform a cross-validation, in our case we tune only on few possible hyperparameters (recall that we will have just 2928 letters) but the method is written in such a way that it's possible to tune also on bigger set of hyperparameter.

During the prediction phase, each tree in the forest independently classifies the input data point, and the final class prediction is determined through a majority vote among the trees, to check this you have to read the code of the RandomForest class of Accord.net.

The loss function that we will use is the 0-1 loss function, also known as the binary classification loss, it measures the error rate of the classifier by assigning the value 1 at the correct predictions and 0 at the incorrect ones. The goal of the algorithm is to minimize this loss function, meaning that it aims to make accurate predictions that lead to a minimal number of misclassifications.

To build the forest, the algorithm goes through the following steps:

1) data subsampling and feature subsampling: random subsets of the training data are selected for training each decision tree, random subsets of the features are chosen for each tree, promoting diversity and reducing correlation between trees;

2) decision tree training: Each tree is trained using its assigned data subset and feature subset;

3) prediction and aggregation: During the prediction phase, each decision tree independently classifies the input data point. The final prediction is made by aggregating the individual tree predictions through a majority voting process.

In the mean time we also perform the cross-validation following the steps:

1) data partitioning: the original dataset is randomly divided into k roughly equal-sized subsets;

2) iteration loop: the cross-validation process involves k iterations, in each iteration one of the k subsets is used as the validation set, and the remaining $k-1$ are used as training set;

3) model training: at each iteration, the model is trained on its training set which consists of $k-1$ subsets, then it's validated on the subset that was set aside for validation in the current iteration. This provides an estimate of the model's performance on unseen data using the 0-1 loss function;

4) results: after completing all k iterations, the performance metrics obtained from each iteration are used to find the best combination of the hyperparameter within the set of the initial hyperparameter values;

4 Choose of the feature: a psychiatry reason

The choice of the features is based on the previous analysis made by Modena's SPDC quoted at the preface section. Considering the statistical test contained in Reami's master thesis and other evidence [4][7], we have that the features written in boldcase are statistically significant and so they can be useful to predict the revolving door behaviour of a patient. On the other hand, the remaining features (e.g. season of discharge, days of TSO, admitting motivation) aren't significant or can be better explained using the previous ones, we have chosen than to remove this features since they can lead to overfitting. It's been noticed, for example, that the admitting motivation usually bring less information than the discharge diagnosis, that's because in the first case psychiatrists have to provide a quick justification to a new patient using usually few informations, while in the second one they have a deeper knoledge of the same person. That's the reason why psychotic decompensation and state of excitement are the admitting motivation that appears the most: they are used as generic purpose when it's impossible to have a clear and quick comprehension of the disorder.

We have checked that the chosen features aren't redundant: we have computed the correlation matrix for the continuous features and the correlation matrix for the drugs utilized as therapy. In Table (1) it's possible to notice that every feature has little correlation with the other one and hasn't a null correlation with the revolving door behaviour. In Table (2) we notice that there isn't a pair of drugs which are almost everytime used together or disjoint, so it's not possible to remove any of them to reduce the overfitting risk.

	hosp. counter	age	days of hosp.	revolving door
hosp. counter	1			
age	-0.07	1		
days of hosp.	0.08	-0.02	1	
revolving door	0.61	-0.11	0.10	1

Table 1: Correlation matrix for the continuous variable.

	NL	LAI	AD	Mood stab.	BDZ	Other
NL	1					
LAI	-0.12	1				
AD	-0.05	-0.22	1			
Mood stab.	0.03	-0.01	-0.02	1		
BDZ	0.10	-0.07	0.01	-0.03	1	
Other	0.07	-0.10	0.08	0.06	0.05	1

Table 2: Correlation matrix for the drugs used as therapy.

Moreover, we wanted also to check that the discrete features are well distribuited on our dataset, we produced Figures (2) and (3) where it's possible to

notice that this happends almost always with few exception. We have chosen to keep our choise anyway since we prefere to follow the meaningful definition used in the dataset that is based on a medical knowledge that we don't have in this moment. It may be a risky choise to join some of the entries without a deep knowledge of the underlying reality.

After all this checks we can conclude that we don't have any reason to change the features chosen for training our model.

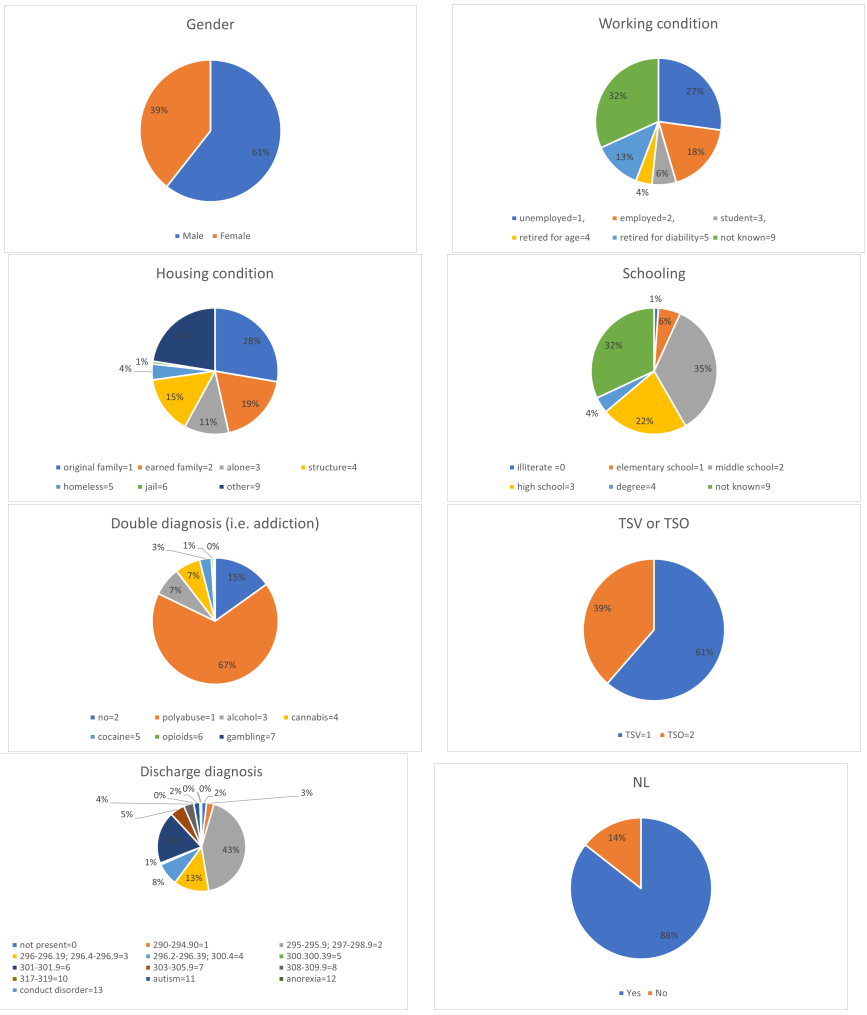

Figure 2:

5 *Project.sln*: a *solution* for our problem

The code we wrote is fully contained in *Project* namespace, it contains three classes: *Program*, *Letter* and *Dataset*. The first one is a static class containing the methods:

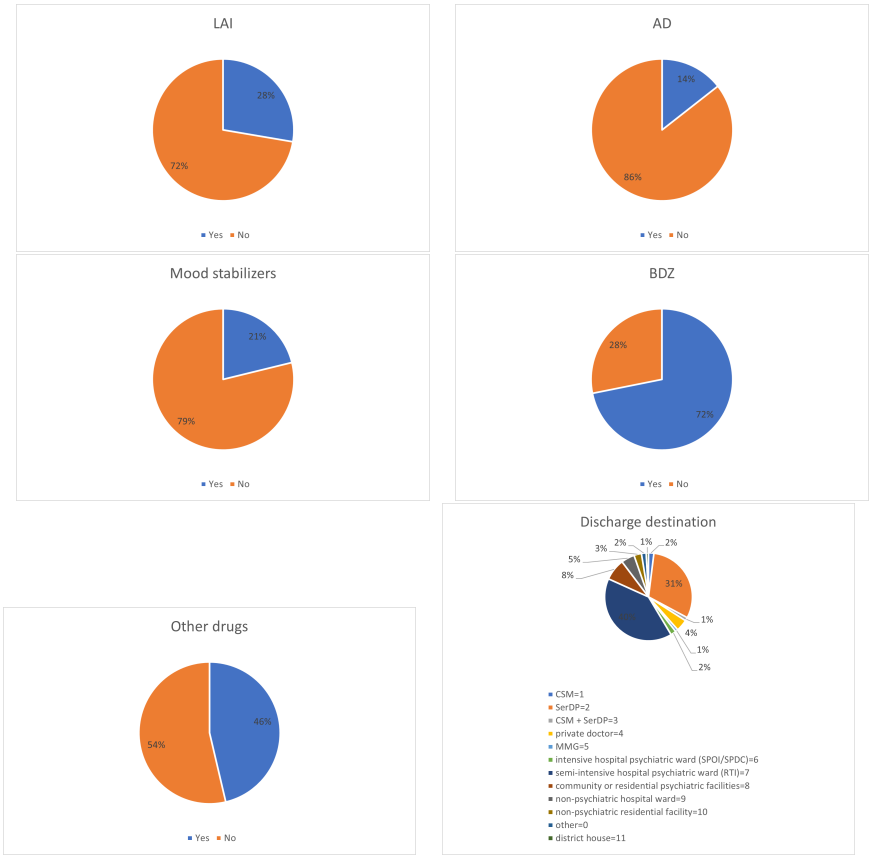
LAI
28%
72%
Yes
No
AD
14%
86%
Yes
No
Mood stabilizers
21%
79%
Yes
No
BDZ
28%
72%
Yes
No
Other drugs
46%
54%
Yes
No
Discharge destination
31%
1%
4%
1%
2%
2%
1%
2%
3%
5%
8%
CSM=1
SerDP=2
CSM + SerDP=3
private doctor=4
MMG=5
intensive hospital psychiatric ward (SPOI/SPDC)=6
semi-intensive hospital psychiatric ward (RTI)=7
community or residential psychiatric facilities=8
non-psychiatric hospital ward=9
non-psychiatric residential facility=10
other=0
district house=11

Figure 3:

- `Main()`: it's able to find the dataset, read it and build the object *Dataset*, then it performs a cross-validation over the hyperparameters `Numbtree`, `Ratio` and `Coverage` and choose the best combination of the tested value using the `ZeroOneLoss` class on the validation set. Once it has concluded the cross validation it finally perform the train on the whole training set and test the final model with the unseen data of the test set. Since we are considering a classification problem, it's also computed the confusion matrix, the accuracy, the error rate, the sensitivity and the specificity. The Main it's able to catch exceptions throwed during the previous steps and in particular the one that may be throwed during the construction of the *Dataset*.

- `TryRandomForest(int32, double ,double, int32[][], int32[], int32[][], int32[])`: it's usefull for test a model during the cross validation. It computes the model, its 0-1 error on training and test sets and the time elapsed.

The other two classes are used instead to define the new objects type *Dataset* and *Letter*, the first one represents the idea of a dataset and it's made of object *Letter* since, in our case, the dataset is a collection of discharge letters. A Dataset object has the fields:

- `Path`: a string containing the absolute path to the dataset.

- `NumbLetter`: an integer containing the total number of letters. We use a counter in the dataset class instead of a static one in the Letter class since we assume it as a characteristic of the whole dataset and in this way it's not needed to call the getter method every time we need this value.

- `Numbfeature`: an integer containing the number of feature used.

- `MatrixLett`: an array of `Letter`s containing all the letters.

- `TrainingRatio`: a double containing the ratio of letter to use for training/test.

- `TrainingSet`: an array of `Letter`s containing the letters used for the training.

- `TestSet`: an array of `Letter`s containing the letters used for the test.

- `IndexTraining`: an array of int containing the row indexes of the letters contained in the training set.

- `CrossValidation`: a jagged array of `Letter`s that will used for cross-validation.

- `NumbOfSuddivisions`: an integer containing the number of cross-validation suddivision.

The methods of this class are:

- `Dataset(string, byte[], byte, double, int32)`: the constructor, it initializes all the fields and in particular selects randomically the letters that will be used for the training set. It can throw, in addiction to the

usual ones, an exception if the number of indexes isn't 17, this is a simple way to check if the number of input indexes are right. Moreover this method is able to manage the presence of missing or incompatible data on the dataset, in these cases it informs the user with a message on the console and removes all the unusable letters. (Notice: there are two dataset in the folder, one with all the letters and one without the unusable one, it's possible to run the code using both but the first one will produce several messages at the console.)

- `ToJag(Letter[])`: private method, coverts the array of letters into a jagged array.
- `ToJagTraining()`: returns the jagged array of the training set.
- `ToJagTest()`: returns the jagged array of the test set.
- `ToJagCrossValidationTraining(int32)`: returns the jagged array for the cross-valiation where it's been removed the subset with the input index. Useful for train in cross-validation.
- `ToJagCrossValidationTest(int32)`: returns the jagged array for the cross-valiation with only the subset with the input index. Useful for test in cross-validation.
- `Label(Letter[])`: private method, returns an array with the labels of the input array.
- `LabelTrainingArray()`: returns the labels of the training set.
- `LabelTestArray()`: returns the labels of the test set.
- `LabelCrossValidationTrainingArray (int32)`: returns the labels for the cross-valiation where it's been removed the subset with the input index. Useful for train in cross-validation.
- `LabelCrossValidationTestArray (int32)`: returns the labels for the cross-valiation with only the subset with the input index. Useful for test in cross-validation.
- `Description()`: prints a description of the dataset. It doesn't return a string since it would be very big, that's why it's not been overrided on the ToString method.
- `GenerateRandomIndex (int32, int32)`: private static method, returns an array containing permuted indexes. It's been used to select the letters for the training set.

On the other hand, a Letter object has fields:

1) `Sex`.
2) `Age`.
3) `Work`.
4) `Family`.
5) `School`.
6) `ExitDia`.

7) `Drug`: addiction.

8) `TSO`: (TSO or TSV).

9) `AfterDim`.

10) `Days`.

11) `Nl`.

12) `Lai`.

13) `Ad`.

14) `Stabil`.

15) `Bdz`.

16) `OtherMedicine`.

17) `OtherLet`.

18) `Dim`: number of fields.

19) `Label`: revolving door behaviour.

The method of this class are:

- `Letter (string)`: constructor method, it throw an exception of type `Exception` if there is a missing value. We chose to use the generic exception since we don't need to distinguish other cases. The data in the string must appear in the order used on the previous list.
- `ToArrayWithoutLabel()`: returns an array containing the fields' value.
- `Description()`: prints a description of the letter. As before, it doesn't return a string, that's why it's not been overrided on the ToString method.

6 Results: a good predictor

Running the code with the cross-validation options given by: {10, 20} for `Numbtree`, {0.8, 0.6, 0.45} for `Ratio` and {0.9, 0.7} for `Coverage`, we obtain that the best combination of the hyperparameter is `Numbtree`=10, `Ratio`=0.6 and `Coverage`=0.9. Training the model with this values we end up with the following confusion matrix:

true positive (TP)= 36	false positive(FP)= 15
false negative(FN)= 13	true negative(TN)= 375

and then we have that

$$\text{accuracy (error on test set)} = 93,62\% \quad \text{error rate} = 6,38\%$$

$$\text{sensitivity} = 73,47\% \quad \text{specificity} = 96,15\%,$$

for completeness we report here also the error on the training set which is 0,48%. We know that it can not be used as an error estimate of the model, instead it can be used to understand if our model is overfitting the data, but this is probabilly not true since we have a really low error both on training and test sets.

Clearly the reported values aren't constant over the iterations because Accord's methods use random generated number, that's the reason way even the tuned hyperparameter my change during the iterations. Anyway it's possible to train 100 different models with the same hyperparameter using the commented method `Run100Forests`, we get from this models the mean values:

true positive (TP)= 34.77	false positive(FP)= 16.37
false negative(FN)= 14.23	true negative(TN)= 373.63

and then we have that

$$\text{accuracy} = 93.03\% \quad \text{error rate} = 6.97\%$$

$$\text{sensitivity} = 70.96\% \quad \text{specificity} = 95.80\%.$$

7 Conclusions

Looking at the mean values of accuracy, sensitivity and specificity it's clear that our model can produce good predictions on unseen data. In particular it's important to notice that it has a good specificity while the sensitivity is good but not so high. From this evidences it may be possible to use the model together with a psychiatric opinion to estimate the future revolving door behaviour of a patient with a good accuracy.

We want now to highlight anyway some restrictions of this project: first of all the dataset was based only on the patients of Modena's SPDC and it means that most of them were living in a precise and restricted geographical region, also the dataset wasn't so big and during the studied six years there was a lockdown that has probabily played a role for the mental health of some people. This considerations end up with the possibility that this results may be not generalizable, anyway, despite all this restriction, we have shown that this type of machine learning can lead to some useful results on this research field: it can be used with bigger dataset to achive a better treatment of the patients, possibly leading to a more appropriate follow-up and a better quality of their life.

We recall here that we tried to recover also which features were the most meaningful for the decision trees but there isn't a method on the Accord libraries able to perform it. It would be really useful, for future research, to investigate also this aspect using other libraries.

Refernces

1. Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001).

2. Di Giovanni P, Di Martino G, Zecca I a. L, Porfilio I, Romano F, Staniscia T. The Revolving Door Phenomenon: Psychiatric Hospitalization and Risk of Readmission Among Drug-Addicted Patients. Clin Ter. 2020;171(5):e421–4.

3. Di Lorenzo R, Sagona M, Landi G, Martire L, Piemonte C, Del Giovane C. The Revolving Door Phenomenon in an Italian Acute Psychiatric Ward: A 5-Year Retrospective Analysis of the Potential Risk Factors. J Nerv Ment Dis. September 2016;204(9):686–92.

4. Donisi V, Tedeschi F, Wahlbeck K, Haaramo P, Amaddeo F. Pre-discharge factors predicting readmissions of psychiatric patients: a systematic review of the literature. BMC Psychiatry. 16 December 2016;16(1):449.

5. D'Orta I, Herrmann FR, Giannakopoulos P. Determinants of Revolving Door in an Acute Psychiatric Ward for Prison Inmates. Front Psychiatry. 2021;12:626773.

6. Friedman I, Lundstedt S, Von Mering null, Hinko EN. SYSTEMATIC UNDERESTIMATION IN REPORTED MENTAL HOSPITAL READMISSION RATES. Am J Psychiatry. August 1964;121:148–52.

7. Gobbicchi C, Verdolini N, Menculini G, Cirimbilli F, Gallucci D, Vieta E, et al. Searching for factors associated with the «Revolving Door phenomenon» in the psychiatric inpatient unit: A 5-year retrospective cohort study. Psychiatry Res. September 2021;303:114080.

8. Ho, Tin Kam (1995). Random Decision Forests (PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282. Archived from the original (PDF) on 17 April 2016. Retrieved 5 June 2016.