BigFive5q Predictive Test

Juan Eloy Suarez - PH125 Data Science - Harvard

30/June/2021

1 Executive Summary

This project uses prediction techniques to simplify the results obtention for the well-known "Big Five" personality traits determination test. The goal is to reduce as much as possible the input from a given user so we can predict final score as accuratedly as possible. Thus, less length, effort and time of test resolution wil probably make it easier to be integrated in other processes where estimation of personality characteristics of an individual can be useful.

This test is made of 50 questions grouped in five *personality traits*. All questions must be answered from 1 (Strongly Disagree) to 5 (Strongly Agree), being some of them "reverted" (formulated as negative). The percentile of the average of each 10-questions group is the final score for that trait. Our **goal is to use only five questions to predict all 45 others and therefore final scores for each trait**, benefiting therefore of such a very light approach to get results.

Measurement of the accuracy of the model will require to define and calculate indicators of how good the model works in predicting the right trait score for an individual. The fact result is a set of five numbers "percentiles" will make us define metrics for assessing how close predictions are to real results.

There are some clear **premises we will follow for this project**:

- Measurable: Strict measurement of results, based on a clear definition of the accuracy obtained.
- Usuful: Prepare model to be usuful for further developments/integrations, i.e. make it as a *pluggable module* to be integrated in other systems. For example, instead of showing always "optimal" questions to be answered, we prefer to make them -partially- random, so a deployment at bigger scale might benefit of variability.
- Indepentent validation: Never use validation (test) data partition to estimate parameters of select any criteria during training process
- Computability: project must be deep in calculation but affordable within a local environment in terms of performance. This means that we will reduce our dataset during some stages, but a wider execution can be always an option without need of changing source code, just the parameter.

There are several **original aspects** on this project:

- Use Recommender: To take advantage of a standard "recommendation model", adapting some concepts to use classical "User x Item" as inspiration for a "User x Question/Trait" structure. The advantage is to use strong prediction tool (i.e. Recommenderlab, etc.), but on the other side fact like the lack of matrix sparcicity or the very "vertical" shape of the ratings matrix (approx. 10^6 users x only 50 columns) require specific attention during the model implementation.
- An app to play: To create an interactive, open access, graphical app tool to facilitate understanding of the concept and even disclosing some of the key prediction steps followed.
- Questions selection: Our own algorithm to optimize selection of questions to get answer for, so the change (to ensure varibility on data retrieval), whilst optizing combiantion so final accuracy is best.

The Big Five personality traits, also known as the five-factor model (FFM) and the OCEAN model, is a taxonomy, or grouping, for personality traits. When factor analysis (a statistical technique) is applied to personality survey data, some words used to describe aspects of personality are often applied to the same person. For example, someone described as conscientious is more likely to be described as "always prepared" rather than "messy".

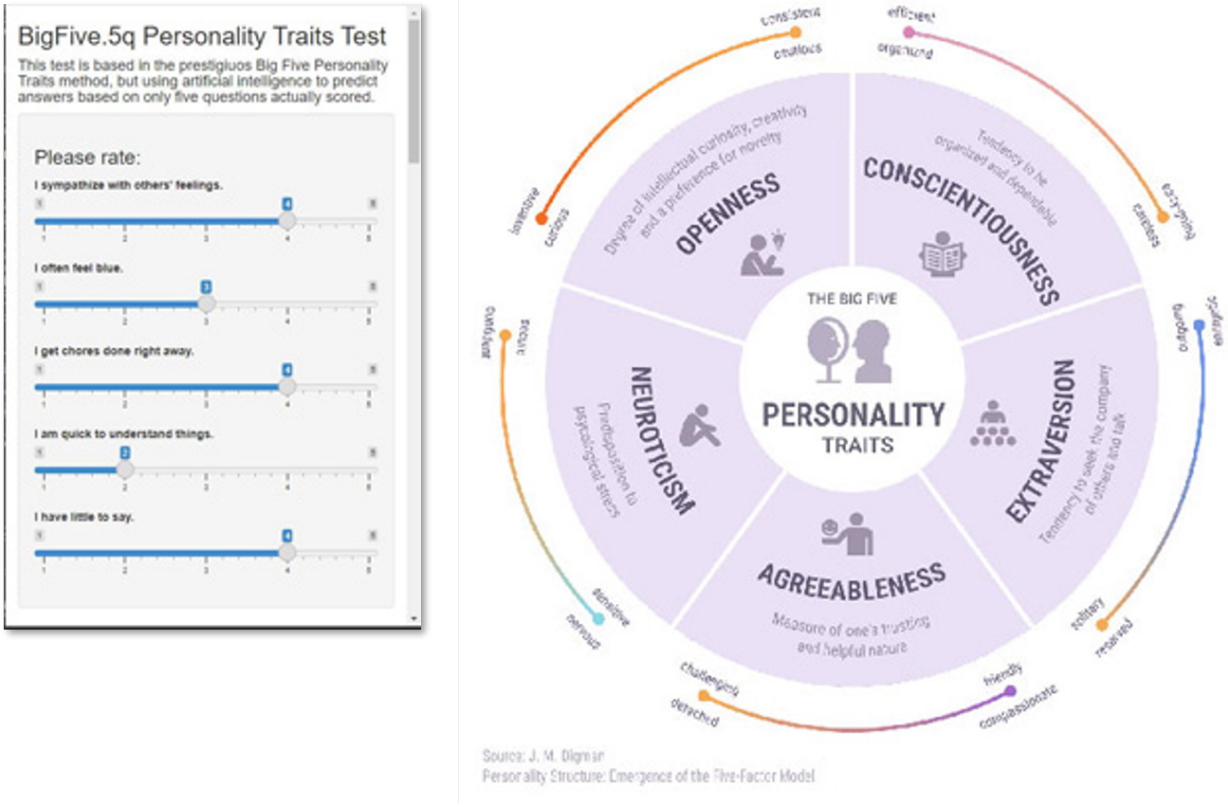

Figure 1: Big Five infography

This theory is based therefore on the association between words but not on neuropsychological experiments. This theory uses descriptors of common language and therefore suggests five broad dimensions commonly used to describe the human personality and psyche.

2 Exploratory Data Analysis

This project is based in the "Big Five Personality Test. 1M Answers to 50 personality items, and technical information" dataset. These data were collected (2016-2018) through an interactive on-line personality test and it contains:

- 1015341 records (tests).
- 110 columns of data, 50 of them answers (1=Strongly Disagree, 3=Neutral, 5=Strongly Agree) of test questions, and all others containing country, location, load date and load/response time for each question.
- A codebook detailing literal wording of each one of the 50 questions.

The personality test was constructed with the "Big-Five Factor Markers" from the International Personality Item Pool (https://ipip.ori.org/newBigFive5broadKey.htm). Participants were informed that their responses would be recorded and used for research at the beginning of the test, and asked to confirm their consent at the end of the test.

For our approach to the model we will discard country, location, and load/response time to questions, so we focus directly on the questions and their answers. As we will remark in the Conclusions chapter, there are interesting possibilities for developing the project using these data, but they are out of the scope of this project. This reduction in the number of prescriptors will significantly reduce the size of our dataset.

Data load strategy

Data are located in Kaggle site (https://www.kaggle.com/tunguz/big-five-personality-test/download). Data size in .CSV is 396 MB, so we will, as a previous step, load, select and convert them to .rds, so a very significant size reduction is obtained to 21 MB full dataset. The code to connect directly to Kaggle server requires an API and some specific coding, so for simplification, we have previously downloaded data locally and saved to a .rds file in directory ./data-source. Code to perform this preprocess and storage is available:

```
# Read Kaggle dataset copied to local directory
#
# Answers to test data
# tab delimited, and  our int columns are character, perhaps because of the nulls!
df <- read.csv("./data_source/data-final.csv", sep="\t", stringsAsFactors = FALSE, na.strings=c("NA","Na
# We select relevant columns to use
df <- df %>% select(c(1:50), dateload, country) %>% rownames_to_column('userId') %>%
  mutate(Month = month(dateload), Year = year(dateload)) %>%
  select(-dateload)
# Remove rows containing any NA
df <- na.omit(df)
# Remove any value not between 1 and 5 in the answers columns
df <- df %>% filter_at(vars(2:51), all_vars((.) %in% c(1:5)))
#
# Let's also load up the questions from the data dictionary
dictionary <-
  read_table("./data_source/codebook.txt", skip=5) %>%
  separate(1, sep="\t", extra="merge", into=c("ID", "Question")) %>%
  data.frame() %>% top_n(50)

#########################################################
# Save .RDS input data to local files
#########################################################
```

```
saveRDS(df, "./data_source/BFtests.rds")
saveRDS(dictionary, "./data_source/dictionary.rds")
```

However, it is not necessary to run the this preload to run project, since all two referred input datasets are already, and publicly, available:

./data-source/BFdata.rds

./data-source/dictionary.rds

Distribution of scores per trait

We will refer as *Scores* to the result (in percentile) of the average of answers (1 to 5 as possible values) corresponding to the ten questions forming each trait. This must be done considering that part of the questions are formualted "reverted" according to published test definition.

Since this type of scoring is very frequent in psychometrics, we will take advantage of the package **psych**. Its function **scoreItems** will directly calculate scores (i.e. mean per group taking sign into account). Detailed documentation on this function can be found: See: https://www.rdocumentation.org/packages/psych/versions/2.1.3/topics/scoreItems

We now visualize how some of the traits tend to be valuated higher than others. For a global analysis, a normalization should we adequate, but we are now focusing in individual study of questions & traits, so the most significant interpretation is that each one of them is approxmimately normal, which be usuful later for percentile calculations.

Test Reliability in each personality trait (internal consistency)

We are now interested in assessing the internal consistency of the test, i.e. to know at what extent our questions are related, behave similarly, when they belong to same group/trait.

The Cronbach's alpha coefficient measures reliability, or internal consistency, to see if multiple-question Likert scale surveys are reliable. Cronbach's alpha will tell us how closely related a set of test items are as a group. More details on Cronback's alpha can be found here: https://www.rdocumentation.org/packages/psych/versions/2.1.3/topics/alpha

According to this coefficient generalized interpretation, values of alpha between 0.8 and 0.9 are considered "Good Internal Consistency", while alpha greater than 0.9 means an "Excellent Internal consistency" situation.

We easily visualize that, despite of some differences among traits, all of them fall in an, at least, "high integrity" interval:

Correlation of the answers to questions

A key aspect for understanding how some questions relate to other, even belong to a different group, is their correlation. It is intuitive to think in correlation of questions as a clear indicator of how unknown values (unanswered question) can be predicted based in known answers to other questions that usually correlate.

So, let's calculate correlation matrix among all questions to visualize global data pattern. Thus, we have also a have a good source for further calculations on what questions must be chosen to answers when there is a possibility to do it. Note some questions are reverted, but this is not relevant for this exploratory analysis, since it affects only to the sign of the correlation, not its absolute value.

Correlation within same group: Questions explicitly associated to each "trait" strongly correlate, i.e. all of them are related because all of them explain the final value of the score for its specific group (trait). We see different traits (groups). Correlation is clearly high within same group.

Regarding behavour among groups, we see also some other **significant correlations** (and of course also lack of them) of some questions with questions that "belong" to different traits. Since our challenge is precisely to use few (only five) questions to explain as mach as possible of the result for all traits, it will be useful to use these correlations to select what specific questions we show to get answer.

3 Methods and train

3.1 Strategy, process

Our challenge is to predict the answers of 45 questions based on the answers to 5 questions we know. Total number of questions is exactly 50, so our matrix of users versus item (questions) is fully filled and we find no *sparcity*. We also observe that the shape of this matrix is pretty vertical, with around rows by only fifty columns. Despite of this singularities, this model seems to reasonable fit a "recommendation system" approach. We will pursue on that and take advantage of the already prebouild strong tools existing in R, specifically the Recommenderlab package.

Nevertheless, some preparation steps will be necessary to *adapt* our problem to the functionalities the standard recommender tool provides.

- Firstly, we need to prepare data for modeling, taking into account that our recommender algorithm will ignore **reverted effect on questions**, so we need to invert their existing answers in advance to make data homogeneous. Also, the **size of base dataset** is excessive in terms of computability in a local environment so we will make a random reduction of it - due to the high density of our Users x Questions matrix, this size reduction will not have dramatics effect in the decrease of our results, though of course could be reconfigured if more powerful systems would run the model.

We will also, in support of our interactive app, to extract data to make it **available within de shinyapps environment**.

A key step will be then to separate dataset in two (**train and validation**) partitions for fulfill machine learning paradigm. For clarity, we will not use *scheme* options of recommenderlab package but to directly build, manage and measure these partitions.

Once data are ready, training and prediction and measurement process starts. First, we need to clearly define **what is and how to measure accuracy** of our model, as easily as possible, taking into account the relative complexity of the calculation of the five resulting scores. A very important step of the modeling will be to build an **algorithm to select what five questions we want to retrieve** answers from. This process must respect a random base to ensure eventual variability in results capture, but optimizing the choice to improve model accuracy. Once done all above steps, we will **run our recommender algorithms**, doing training of each model, prediction, calculation of final scores and storing results for comparison.

3.2 Data preparation for modeling

Reverted questions treatment

Our data contains answers to questions considered "reverted". This means that the question is written in a negative way and, for global analysis, must be scored reverting results recorded. For example, questions "I am the life of the party" and "I don't talk a lot" belong to same trait "Extroversion". However, first second one is reverted, so for direct comparison we need to convert 1 to 5, 2 to 4, 4 to 2 and 5 to 1 answers.

This conversion is already covered by the generic scoring analysis used for package PSYCH, and just identifying with a minus sign before question Id, function internally reverts answers value. This was used in the EDA part above.

However, we are going to use functions based in recommendation methods (recommenderLab) based in linear algebra and distances. This algorithms use "ratings" always meaning "positive", never "reverted". For this reason, we will, before analysis, revert results of "reverse" questions and when necessary keep using regular PSYCH functions, but marking those questions as regular/positive (removing the minus sign).

We see mean and standard deviation of each trait's questions. This shows us significant differences, as shown during EDA, and will be useful to use normality for estimating probabilities os score (percentiles) as defined by the standard test.

Dataset size reduction

Althugh Exploratory Data Analysis was done using all data available, we are now going to randomly reduce data size. This is done for computability and is defined in a single parameter *nObservsDevelopment* that can be increased at anytime if system capacity allows.

Save input data for shiny app

The shiny app is an interactive implementation of this model. It simplifies some parts of the process and shows prediction generated for each questions, as well as the calculated scores. Its purpose is educational and as a demo, it does not store data nor submit specific transaction, i.e. allows simulation with different values.

For this version of the web app we pass reduced version of the dataset to allow online training and prediction. There are significant improvements that could be added to the tool, whick are remarked in the *Conclusions* chapter of this document.

Separate data in partitions

After data exploration, cleanup, adjustment, and reduction, we will separate in two partitions so we can separately train and test our models. We have the option to use *built-in* partition, testing and accuracy functions of the package recommenderlab, but for clarity and control, we will manage directly and independently these data partitions.

3.3 Training and prediction

Accuracy measurement: metrics of prediction success

A good definiiton of the accuracy we expect from our model is a key step during development. The multi-result (5 Scores) this test outcomes makes it necessary to define our expectation for the predictions in terms of:

1) What exactly we want to measure for accuracy
2) A reference of how good the prediction is when done randomly

What to measure? The challenge of the project is to predict Scores for each one of the five personality traits, but these scores directly depend on the answers to 50 questions (10 per trait) where we only know real value of five of them. Thus, our accuracy will depend on how close we are to the real Score for each trait, which is originally expressed in percentile. Since this measure can be understood as approximate/soft, we assume that, for a specific trait, hitting (predicting) the correct quartile could be considered as reasonable good result. In similar terms, hitting same half (high-low) is also -though less- a good result. Looking at all five trait as a set, we will also establish as a good result to correctly predict (high-low) most traits, i.e. three or more of the five. So, our accuracy metrics will be defined as:

Accuracy Indicator	Definition
'Hits quartile' for a trait	Score for a specific trait predicts correct quartile (1-25, 26-50, 51-75, 76-100)
'Hits quartile' all traits	average of *Hits quartile* for all traits of an observation (test)
'Hits HighLow' for a trait	Score for a specific trait predicts correct half (1-50, 51-100)
'Hits HigLow all traits'	average of *Hits quartile* for all traits of an observation (test)
'3+ hits HighLow'	Scores of three or more traits predict correctly halves

How good would be a random estimate? (Montecarlo approach) As reference for accuracy improvement, we will estimate what the Scores would be if using just a random criteria for predicting answers to each question. The random distribution of answers to 50 questions with 5 possible answers same probability (1/5) follow a binomial (Bernouilli) pattern. However, due to the calculations required for our accuracy indicators, we will code a **Montecarlo** simulation of a random selection, which will allow us to simulate results. These accuracies will be considered then as "base reference" for further improvements during modeling process.

This function adjusts random score taking into account that answers of one of every 10 question is known. For simplicity, we assume a linear improvement of accuracy for 1/10 and correct by chance of randomly hitting (1/5)

Questions selection: minimum combined correlation algorithm

Our model relies on **only five questions** actually known, out of 50, to predict the other 45. This means the best possible selection of the five questions we get real answer for is very important to better predict the others. However, choosing just the same apparent best combination (always the same for given dataset) seems not to be the most open, realistic, rich option, since in case of moving this model into production, would result in a very poor diversity of input rsults, which long term would drive to worse results and lack of diversity. Thus, we will state as premise that, one of the questions to get real answer from, must be randomly selected (seed question), whilst the other four ones can be generated based on expected best results.

The algorithm we propose to select those other four question can be called "minimum combined correlation method", and consists in selecting those questions having less (absolute) correlation with the seed question and with the other chose questions. We implement therefore a recursive approach based in comparing the average correlation of all pairs of questions for each combination (10) of potential questions including the seed question (10000)

Loop available recommendation algorithms

Our approach for predicting 45 answers (item) of an user that actually answered only to 5, will be to a Recommendation Model, based in the library Recommederlab. Tha basic idea is to consider Users (those who answer) and Items (questions) and apply some of the methods of standard recommendation. However, there are some singularities in our case compared to a *typical movie/book* recommendation case:

- No sparcity: our matrix of answers *Users X Items* has all cells filled, since we have data for all users answers all questions
- Matrix is pretty "vertical" having more that 800000 Users (after initial cleanup) and 50 Item (columns)
- Items have clear correlated components, since questions are grouped by traits

Thus, we will choose adequate algorithms based on singularities:

Due to the shape of the matrix, we discard IBCF (Item Based collaborative Filtering) due to the risk in some cases fo not generating predictions for all item (could be fixed with average/median filling though loosing accuracy). Regarding no scarcity we also discard Popular method.

In order to generate a rich approach for our prediction, we'll focus in a collaborative filtering method (UBCF) and in a matrix factorization method (ALS) - we choose it instead of SVD for its superior accuracy given singularities of our User x Item matrix:

UBCF (User Based Collaborative Filtering): this algorithm mimics word-of-mouth by analyzing rating data from many individuals. The assumption is that users with similar preferences will rate (answer) items (questions) similarly. Thus missing ratings for a user can be predicted by first finding a *neighborhood* of similar users and then aggregate the ratings of these users to form a prediction.

The neighborhood is defined in terms of similarity between users. Popular similarity measures for are the *Pearson correlation coefficient* and the *Cosine similarity* (the one we will use). This similarity measure is defined between two users u_x and u_y as:

$$sim_{\cos}(\overrightarrow{x}, \overrightarrow{y}) = \frac{\overrightarrow{x}.\overrightarrow{y}}{\|\overrightarrow{x}\|.\|\overrightarrow{y}\|},$$

where $\overrightarrow{x} = r_x$ and $\overrightarrow{y} = r_y$ represent the row vectors in R with the two users' profile vectors. $sd(\mathring{u})$ is the standard deviation and $\|.\|$ is the $l^2 - norm$ of a vector. For calculating similarity using rating data only the dimensions (items) are used which were rated by both users.

Now the neighborhood for the active user $\mathcal{N}(a) \subset \mathcal{U}$ can be selected by either a threshold on the similarity or by taking the k nearest neighbors. Once the users in the neighborhood are found, their ratings are aggregated to form the predicted rating for the active user. The easiest form is to just average the ratings in the neighborhood.

$$\hat{r}_{aj} = \frac{1}{|N(a)|} \sum_{i \in \mathcal{N}(\dashv)} r_{ij}$$

ALS (Alternating Least Squares): ALS is an iterative optimization process where, for every iteration, we try to arrive closer and closer to a factorized representation of our original data. We have our original matrix R of size $u \times i$ with our users, items and some type of feedback data. We then want to find a way to turn that into one matrix with users and hidden features of size u x f and one with items and hidden features of size $f \times i$. In U and V we have weights for how each user/item relates to each feature. What we do is we calculate U and V so that their product approximates R as closely as possible: $R \approx U \times V$. By randomly assigning the values in U and V and using least squares iteratively we can arrive at what weights yield the best approximation of R. The least squares approach in it's basic forms means fitting some line to the data, measuring the sum of squared distances from all points to the line and trying to get an optimal fit by minimising this value.

With the alternating least squares approach we use the same idea but iteratively alternate between optimizing U and fixing V and vice versa. We do this for each iteration to arrive closer to $R = U \times V$.

4 The shiny application

In addition to the model created based on existing dataset, we create also an interactive tool to show in practice how prediction works. The tool is fully developed in R as part of current project and provisioned to internet through platform Shinnyapps. It is basically an interactive web page to allow any person to answer 5 randomly selected questions, and based on them, estimated personality traits Bif Five Scores by *predicting* al other 45 remaining question using our recommendation model.

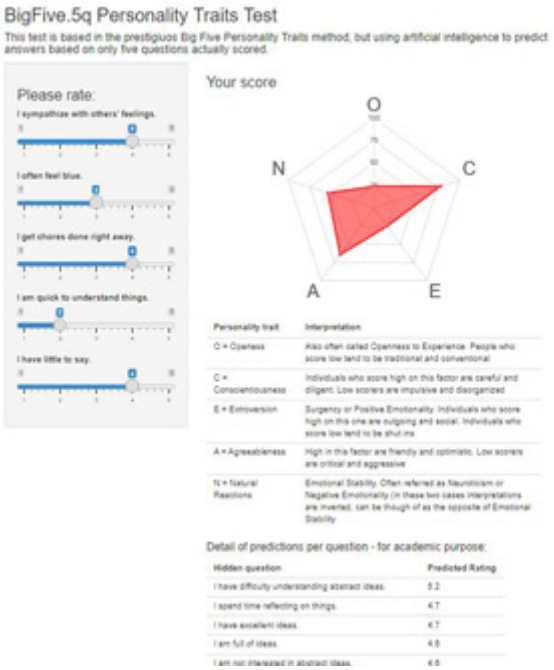

Figure 2: Screenshot1 of interactive tool

For practical and academic reasons, tool is simplified to allow wide interactive testing and avoid any data protection issues .some of the simplification changes:

- An user can repeat test as many times as desired.
- Algorithm implemented is UBCF (second best in our study) instead of ALS for interactivity performance.
- Questions are chosen randomly instead of using our just developed *Minimum Combined Correlation Algorithm*
- Most the most important... all *hidden* predicted question's answers are shown for instructional purposes

This promising tools inspires many developments to incorporate nice functionalities, potentially very useful in combination with other tools for other specific needs, Some if them:

- registrar usuario
- almacenar resultados
- añadir traducción
- crear un "submit" para queda el resultado sea "inamovible" (o explicar lo dejó así "interactivo" y mostrando las predicciones de preguntas con fines académicos +- "disclosed mode")
- añadir otras preguntas de control/clasificación combinadas (...¿mes nacim.? ¿do you like cheese?...) + país + IP + timestamp + vbles1:3

- combianr con outcomes conocidos (por ejemplo ADE-Univ. Comillas)
- prohibir repetición de entradas por el mismo usuario

5 Results

We have got aa appretiable result just with Phase 1 Model, and in case we have event time data available for cases to predict, we will obtain even more improvement on RMSE. As a results sumamry, running the final algorithm of our modelling on the –validation– set yields the following rating:

We can know compare how accuracies, as defined, get significantly improved with both ALS and UBCF methods algorithms respect to base “random” reference generated with the Montecarlo approach

6 Conclusion

We have implemented a model that ...

Finally, we can mention some potential improvements for future versions:

- Optimizar tuning Recommender (evaluationScheme, k-fold, bootstrap, no-sparse,) usando sub-particiones
- Probar aumentando el tamaño del dataset (usando sub-participes de training)
- Ensemble de mejor algoritmo/trait
- Negociar con ADE para crear modelo de rendimiento académico "5q"
- Mantener random de preguntas (para diversificar datos obtenidos) pero mejorar selección de otras preguntas por «correlación ó CART
- Posibilidad de >5 preguntas
- Bias Month, Country (mostrar en EDA)
- Mejoras SHINY: control de no repetición, botónSubmit, añadirPreguntasControl, almacenarResultados_ojoGDPR (IP, hora), precalcularModelo, ...
- Vincular las 5 preguntas a otros juegos de preguntas/prescriptores para obtener in modelo único con outcome OCEAN: 2 ó 3 preguntas "misceláneas" (¿mes nacim.? ¿do you like cheese? ¿...?)

As a final conclusion, we can state we have reached challenge goal in terms of project requirements and accuracy for the validation set provided.