BigFive5q Predictive Test

Juan Eloy Suarez - PH125 Data Science - Harvard

30/June/2021

1 Introduction

Executive Summary

In this project, . . . * Prestigio del test BigFive/OCEAN * Valor de los resultados aproximados VS "pesadez" de 50 preguntas = interés en lograr resultados aprox(*) con sólo 5 preguntas* (*): defino una función de Accuracy HighLow por trait acertando >=3 de 5 preguntas

OBJETIVOS: * 1) minimizar test * 2) hacerlo móvil * 3) preparar para vincular con otros datos * 4) medir y mantener alta accuracy * 5) preguntas seleccionadas. . . afán recopilatorio/constructivo VS random/fijo

ORIGINALIDAD: * Adaptar el modelo de Recomendación (aunque la matriz no tenga sparsity) * Relleno de IBCF (si lo hago) * Versión "disclosed" para móvil * 1) usar recommender; 2) servir de "módulo" para futuras integraciones

PREMISAS: * no usar test para parámetros (uso sub-partición); 2) computabilidad local; 3) respeto estricto al test

2 Understanding the data

Exploration of the . . .

- explicar prescriptores descartados (tiempos de carga, ubicación, etc)
- criterio país
- criterios año y mes
- correlaciones
- gráficos de ratings

3 Methods and train

3.1 Strategy, process

1) simplify prescriptors
2) use questions as items for recommendation
3) calculate OCEAN results
4) measure accuracy
5) optimize questions to show
6) pre-load free fields
7) publish shiny
8) store results

-PARTITIONS -FUNCIÓN ACCURACY (deciles) -INICIAL: comprobar si “reverse questions” afectan al recommender (estudiar impacto en: 1)recommenderPrincipal; 2)recommenderShiny; 3)cálculoOCEANinicial; 3) cálculoOCEANshiny) || fórmula: abs(6-x) -TRAIN: criterio selección preguntas (random? cor?) -TRAIN: criterio # preguntas (5?) -TRAIN: criterio algoritmo a usar (UIBF?) -TRAIN: tamaño óptimo dataset

3.2 The shiny application

-SHINY: registrar usuario, país y timestamp + vbles1:3 -SHINY: almacenar resultados -SHINY: añadir traducción -SHINY: 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”)

Exploratory Data Analysis

Calculate scores for each observation based on its questions

Version AS-IS: We score before change of “reverted questions”

Scoring analysis requires calculation of each trait score based on its question’s answers. Process is to obtain the average within group (taking sign for reverted questions into account), and the obtain the percentile of that value (using as reference the whole population of answers)

Test reliability in each personality trait

Reliability (internal consistency)

Questions of the test are grouped in 5 categories (personality trait). We can expect a high correlation (positive or negative) among the 10 questions in each group. We also expect also reliability alfa vualues. 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. https://www.rdocumentation.org/packages/psych/versions/2.1.3/topics/alpha

Correlation of the answers to questions

We calculate correlation among all questions. Some questions are reverted, but this affects only to the sign of the correlation.

Clean environment before training

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. This fact 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 that use "ratings" allways meaning "positive", never "reverted". For this reason, we will, before analysis, revert results of "reverse" questions and when when necessary keep using rgular PSYCH functions, but marking those questions as regular/positive (removing the minus sign)

We see mean and standard deviation of each trait's questions

Dataset size reduction

Let reduce dataset size to facilitate development. This step is intended to allow agile analysis in locale environment. Previous EDA has been done using all observations

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 tool to work we pass reduced version of the dataseet to allow training and online prediction.

Separate data in partitions Create Train and Test partitions: Validation (Test) set will be 20% of data

Training and prediction

Accuracy measurement: metrics of prediction success

We need to define clearly how our model increases the performance of prediction. The first step is define clear measures of the accuracy it reaches. 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 fro 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:

- "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" for all traits: Scores of three or more traits predict correctly halves

Accuracy measurement: metrics of prediction success

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, out of 50, to predict the other 45. This means the best possible selection of the five questions we get real aswer 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 respect to a typical movie/book recommenadtion case:

- No sparcity: our matrix of answwers 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 assumed correlative 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 Colaborative Filtering): this algorithm which tries to mimics word-of-mouth by analyzing rating data from many individuals. The assumption is that users with similar preferences will rate items 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, either by taking a given number of most similar users (k nearest neighbors) or all users within a given similarity threshold. Popular similarity measures for CF are the Pearson correlation coefficient and the Cosine similarity. These similarity measures are defined between two users ux and uy as

  ¡¡FORMULA!!

and

¡¡FORMULA!!

where ~x = rx and ~y = ry represent the row vectors in R with the two users' profile vectors. sd(·) is the standard deviation and k · k is the lˆ2-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 N (a) SIMBOLO_PERTENECE 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.

¡¡FORMULA!!

- ALS (Alternating Least Squares): ALS is an iterative optimization process where we, for every iteration, try to arrive closer and closer to a factorized representation of our original data. We have our original matrix R of size u x 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 x 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 ~ U x 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.

4 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

5 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: multialgoritmo_ensemble, botónSubmit, añadirPreguntasControl, almacenarResultados_ojoGDPR, . . . -Vincular las 5 preguntas a otros juegos de preguntas/prescriptores para obtener in modelo único con outcome OCEAN

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