Analytics Cup 2019/2020

Classifying Ownership Interests of Physicians in Pharmaceutical Companies

The Challenge

You are given data about licensed physicians in the United States, as well as payments or other transfers of value that they have received from pharmaceutical companies. These payments have been reported to government agencies according to regulatory requirements. This is real-world data, but we have made some modifications to tune the skill level and hardware requirements to the student project. Transactions include things like speaker- or consulting fees, training seminars (e.g. learning to use a medical device), conference travel, food and beverages at business meetings, etc. They may also include payments that are made in connection to some ownership interest that the physician may hold in the pharmaceutical company, e.g. dividend payouts, payments in the form of stock or stock options, etc. You are given 1 400 000 instances of transactions which have been received by 6000 physicians in the years 2013 to 2019 (transactions.csv), as well as additional data about these doctors (physicians.csv) and companies (companies.csv). For 5000 of these doctors, transactions related to ownership interest (if any) are included, for the other 1000 (the *test set*), these kinds of transactions have been removed from the transaction data. For each of these 1000 physicians, your model must make a prediction about whether they have had such ownership interests (prediction=1) or not (prediction=0).

Definition of Ownership Interest

'Ownership interest' in the context of this challenge is defined as follows: ***A physician has an ownership interest if he or she has received one or more transactions from a pharmaceutical company in the relevant time frame which has been reported to the government as being related to an ownership interest.*** *In the training data, such transactions are marked by the corresponding indicator variable `ownership_indicator`, but for the 1000 doctors in the test set, these transactions will not appear in your training data.*

Evaluation

Your predictions will be evaluated based on the performance measure of **balanced accuracy** – the arithmetic mean of Sensitivity and Specificity.

<table>
<tr><td></td><td colspan="3">Truth (physician had at least one ownership interest)</td></tr>
<tr><td rowspan="3">Your Prediction</td><td></td><td>YES</td><td>NO</td></tr>
<tr><td>YES</td><td>True Positive</td><td>False Positive</td></tr>
<tr><td>NO</td><td>False Negative</td><td>True Negative</td></tr>
<tr><td colspan="2" rowspan="2"></td><td>Sensitivity = True Prositive Rate
= TP/(TP+FN)</td><td>Specificity = True Negative Rate
= TN/(FP+TN)</td></tr>
<tr><td colspan="2">Balanced Accuracy = BAC = (Sensitivity + Specificity) / 2</td></tr>
</table>

The Data

Train and Test Dataset – Physicians

The main dataset (physicians.csv) contains information about 6'000 physicians in the United States.

Column	Description
Physician_ID	(Integer) A unique identifier for the physician
Set	(String) either 'test' or 'train', indicating whether any matching ownership interest transactions will be included in transactions.csv ('train') or whether you will have to predict their presence ('test')
First_Name	(Strings)
Middle_Name	
Last_Name	
Name_Suffix	
City	(Strings) Place of the physicians' primary practice/business
State	
Zip_Code	
Country	
Province	
Primary_Specialty	(String) The primary field of the doctor.
License_State_X [1-5]	(String) The state in which the physician is licensed to practice medicine. (In case of licenses in multiple states, up to 5 are included.)

Payments Data – transactions.csv

Column	Description
Record_ID	(Integer) A unique identifier for the payment
Physician_ID	(Integer) The id of the recipient of the payment or transfer of value
Company_ID	(Integer) The id of the company making the payment or transfer of value
Total_Amount_USD	(Double) The total value of the transaction in USD
Date	(Date) Date of the payment (first payment if referring to multiple)
Number_of_Payments	(Integer) Number of payments included in total amount if paid in multiple transacitons
Form_of_Payment	(String) the method of payment used, e.g. cash, in-kind items, services …
Nature_of_Payment	(String) The nature (i.e. reason) of the payment
City, State, Country _of_Travel	(Strings) If company paid for travel of the recipient, indicates the destination
Ownership_Indicator	(String) "Yes" marks transactions related to an ownership interest of the Recipient (or a close family member) in the Company making the payment. *(Such transactions have been removed for physicians in the test set!)*
Third_Party_Recipient	(String) Indicates whether the transactions was received by a third party on behalf of the physician
Charity	(String) Whether such a third party is a charitable organisation
Third_Party_Covered	(String) Whether such a third party is covered by the same transaction disclosure rules as the physician
Contextual_Information	
Related_Product_Indicator	(String) "Yes" if the transaction is directly related to a specific product, e.g. a drug, medical device, etc., that the company is marketing to the doctor, etc.
Product_Code_X,	(String) Additional information about the related product(s), if any. Up to 3 related products may be included in an instance.
Product_Type_X	
Product_Name_X	
Product_Category_X	

Pharma Company Data – companies.csv

Column	Description
Company_ID	(Integer) A unique identifier for the company
Name	(String)
State	(String)
Country	(String)

The Rules

Submissions

A valid submission contains of a csv-file containing predictions and a script that generates these predictions from the data that you have been given. Your submitted script **must be self-contained and reproducible**, more on that below. Your prediction file will be graded automatically and judged based on the performance measure of **balanced accuracy** it achieves on the test set. Your team can make **up to 10** valid submissions. Only the ***best valid submission*** from your team will be evaluated for grading.

We have provided you with a sample submission file (with entirely random predictions) which you can use to check whether the format of your generated submission is correct.

Prohibitions

The following things are strictly prohibited and will result in disqualification:

- You may **NOT** hard-code predictions for any instances in the test set. All predictions must be based on your model output.
  This applies both to individual predictions (i.e. **forbidden:** prediction[id==200001] <- 1)
  as well as to fixed rules (**forbidden:** prediction[State==CA] <- 0).
  *Note: Hard-coding **features** to be used in the model is generally allowed.*
- You may **NOT** work together with other teams. If we find that you copied work or cooperated, both teams will be disqualified.
- Needless to say, you must **NOT** try to reverse engineer the private training set based on the original dataset underlying this challenge. (Even if you have access to the original data, we can promise you that this wouldn't be an efficient use of your time anyway.)

If you are unsure about whether something is allowed or not, please reach out to us or ask in the moodle forum! In cases of ambiguities, we reserve final judgment on whether a given submission violates the rules above!

Reproducibility

All submissions must be **reproducible**, i.e. the submitted R script must reproduce the same prediction file, even when run on a different machine at a different time. To ensure this, your scripts should (at least) follow the following guidelines:

- Import all packages that you use **at the very top** of the file. If you implicitly use a backend package via tidymodels/parsnips (`set_engine`) (or via an mlr-learner, etc), please explicitly import the library anyway, or, at a minimum, add a comment to the top of your file.
- At the top of your script, right after the imports, set **`set.seed(2021)`** to seed R's random number generator (rerunning the script will then give you the same results in random operations). Some machine learning packages (such as h2o) manage their own random number generator that's not managed by R. If you use such packages, set the seed in the same manner.
- Do NOT change the file names of the training and test data sets. Your script should `read` the files (and write submissions) from/to ***its own directory.*** *(i.e. `read_csv('payments.csv')`, rather than `read_csv('C:/Users/name/my_files/more_directories/I_renamed_the_payments_file.csv)'*
- Do NOT modify the content of the data files provided. All data preparation should happen within the provided script.
  You may want to save intermediate results that took a long time to generate (data, models, etc.) to disk and read them again. That is fine for prototyping, but not for the final script you submit.

The following last point will not be handled as strictly but you should nevertheless adhere to it:

- Your submitted script should be a (reasonably) minimal implementation to generate your model. We don't expect you to spend any time on optimizing this, but please use good judgment to avoid unnecessary computation in evaluation.
  Example 1: *To find your perfect model, you performed a hyper-parameter search that took 3 days to run. Your submitted script should then only train your final model using the (hard-coded) final hyperparameters that you found. Don't include the search in your file.*
  In such a case, add a short comment about how you arrived at the hyperparameters (or comment out the code for the search)
  Example 2*: You trained 20 models and decided on your favorite one to create a submission at the end. Your submitted script should* ***only*** *trigger training of your favorite model, not all 20 models. (Delete or comment out the code for the other models in your submission.)*
- Although good solutions should be possible in <<10 min runtime on modest hardware (e.g. 5-year old laptops), some groups might have models that take longer to train. If your script takes a very long time to run, please include a comment at the top of your script that includes approximate runtime and info about your computer. (e.g. `#1.5 hours on dual-core laptop with 4GB RAM`).

Frequently Asked Questions

Languages other than R

Some students have asked whether they may use other languages than R (such as python) for the Analytics Cup. This is permitted in general, but we cannot provide you with any support and you must adhere to the same standards of reproducibility found above.
If you want to use python, you must submit a single, self-contained python script. For grading of python scripts, all packages you use must be installable via conda or pip.
If you want to use any language other than R or python, please contact us beforehand.

Jupyter and Rmd-Notebooks

Some students prefer writing code in Rmd or Jupyter notebooks rather than flat .R scripts. Submissions uploaded as notebooks are generally accepted, but must adhere to the same Reproducibility requirements outlined above. Especially, the cells in your notebook must run **in order, from top to bottom,** and should not contain additional exploratory analysis steps, in particular none that take a long time to run. For submissions uploaded as .ipynb files, additionally please strip all cell output from the file before submitting.

Cloud Services, Google Colab etc

The challenge is designed to run fairly comfortably on modest hardware (e.g. 5 year old laptops with ~4GB RAM and dual-core processors). If you want to use cloud platforms to build your models, you may do so, but you will nevertheless have to submit self-contained scripts that can run on our local machine. (You may **not** submit a link to a repository, etc., instead.)