Automated Coding using the IMF's Catalog of Time Series

This report summarizes a pilot study conducted by the Statistics Department of the IMF to automate the coding of time series collected from member countries using the Catalog of Time Series (CTS), an internal nomenclature of descriptors and mnemonics of economic and financial variables used in the policy framework of the IMF. The report highlights the work done during the second phase of the ONS-UNECE Machine Learning project to move the project from a pilot phase to a production tool. For further background on this project, please refer to the IMF report of last year's pilot study.

Report prepared by the data analytics team of the IMF Statistics Department[1]

Authors

Alberto Sanchez, Alessandra Sozzi, Ayoub Mharzi, Lamya Kejji, Marco Marini, and Yamil Vargas.

Version of February 14, 2022

[1] ***This report should not be reported as representing the views of the IMF***. *The views expressed in this report are those of the author(s) and do not necessarily represent those of the IMF or IMF policy.*

Table of Contents

1. The Data

1.1 Background

The IMF Statistics Department assists member countries to implement the IMF's Dissemination Standards Initiatives (SDDS Plus, SDDS, and e-GDDS). These standards require or recommend the dissemination of selected data through the country's National Summary Data Page (NSDP) using the SDMX format. This project used data collected from NSDP of countries participating in e-GDDS, SDDS, and SDDS Plus.

- Example of country file structure

Descriptor	INDICATOR	BASE_PER	2013	2014	2015	2016	2017
Nominal GDP by Activity	?	?	1432669.8984	1480521.3947	1315250.5834	1311248.3355	1405006.8341
Agriculture, forestry and fishing	?	?	9223.0676	9468.2351	9746.3480	10175.8220	10721.0735

- Example of CTS structure

Code	Full Descriptor	Methodology Reference	Sector - Name	Topic - Name
NGDPVA	National Accounts, Activity, Memorandum Items, Gross Value Added, Nominal		National Accounts	Activity
NGDPVAGA	National Accounts, Activity, Memorandum Items, Gross Value Added, of which Government Activities, Nominal		National Accounts	Activity
A_CPC21_0	Economic Activity, Production, By Central Product Classification (CPC) Version 2.1, Agriculture, forestry and fishery products	FAO SEEA AFF; CPC Version 2.1	Economic Activity	Production

1.2 Data Expansion

The first step of our work was to combine indicators descriptors, from the collected country files. The goal is to create **Full Descriptors** comparable to those already existing in the IMF's CTS. This was done by concatenating parent indicators descriptors following the hierarchy in the file ("**parent1 descriptor, parent2 descriptor, … , current indicator descriptor**"). During the pilot study, since our main goal was to test the feasibly of such a solution (using machine learning to generate codes for non-coded indicators) the team focused on using files with a specific structure that simplified the creation of descriptors with the full hierarchy and only using indicators with English descriptors. Our goal for this year's work was to build on the results obtained last year and develop a generalized automated solution for coding economic and financial indicators available in the IMF. To achieve this, we started by looking for additional datasets for which users in the IMF are facing a similar coding issue and incorporate these new indicators during the development of our final solution.

1.3 Feature Categories

The augmented dataset led us also to reassess the features used for prediction. After a review exercise, features common to all datasets were highlighted, which resulted in creating categories for the features used.

- **Mandatory features** that must be available in a dataset.
- **Optional features** that provide additional insights for better predictions, but without which the ML solution can still generate predictions.

Mandatory Features	Optional Features
DESCRIPTOR	TIMESERIES
UNIT_MULT	TIMESERIES_KEYS
SECTOR	BASE_PER
METHODOLOGY	DATA_DOMAIN
	UNIT

This data review work resulted in:

1- An increase in the indicators used to train and test the model from around **40,000** indicators (during the pilot study) to over **115,000** indicators.
2- Better clarity for our internal users, with guidelines on files structures to run code predictions and requirements for mandatory features.

2. Model retraining

As already mentioned, in the pilot study only a sample of countries and a sample of submission files with certain characteristics were used to train and test the models and generate predictions. This led to promising results overall. However some indicators were systematically predicted in the wrong classes and this needed to be improved.

In the new phase, the training dataset was improved to pre-classify some of the worst performing sectors. Such improvements included:

- Identifying the sector (National Accounts, Economic Activity etc.) for each input entry based on the available information.
- Augmenting the embedding space by adding dummy versions of certain features.

However, none of these approaches seemed to improve much on the results from the pilot phase. Following on the work described previously on feature categories, a new approach was implemented based on using the minimum amount of features-engineering using only the 3 mandatory variables (indicator descriptor, sector, and methodology) as a baseline to improve on. In addition, and with the objective of having a functional tool for different users across the IMF, only a minimum amount of pre-processing was adopted during this year's work to ensure reproducibility for slightly different file structures. Some of the characteristics dropped during the data processing in the pilot phase were kept:

- Non-English descriptors.
- Original hierarchy structure of the descriptor.
- All types of submission files were kept (standard and non-standard).

This led to a significant improvement in the results previously obtained and was mainly due to the increase in the data used to train the model (training data was increased 3.5 times).

With this approach**,** we obtained better quality predictions using the same model and feature extraction techniques used in the pilot study (i.e., Nearest Neighbors using cosine similarity with Word2Vex, and R package fastrtext that implements Facebook's FastText library.) The model of choice was "skipgram" with the default parameter configuration:

> *vector dimension: 100, minn = 3, maxn = 6 (i.e., take all the sub-words between 3 and 6 characters, which may have proved useful to handle typos or Spanish descriptors)*

Subsequent model re-trains will be conducted using new labeled data provided by final users of the tool from the list of predicted codes resulting from the current version of the model.

3. Updated Results

The expansion of the dataset used for training (i.e., "the training dataset"), in addition to work done on the feature selection (using only the few common mandatory features), resulted in an overall improvement in the prediction accuracy compared to the pilot phase. As mentioned earlier in the report the augmentation of the embedding space was not a conclusive approach.

Table 1 provides a comparison of the predictions' accuracy across all indicator domains for each approach compared to the pilot phase by data domain. However, to be consistent with standard performance metrics, we also provide a summary of these in Table 2 and Table 3. It is important to note that the results in Table 2 differ from those in Table 1 not just in the aggregation by data domain but also in 2 other aspects:

- Once the model is trained, we compute the predictions, that is, cosine similarities across vector representations, using unique pairs of text and labels (codes). This improves performance and accuracy. For instance, by keeping the original training dataset, multiple repetitions of the same code were returned as top predictions, so, if top 10 results were returned, a wrongly predicted code with 10 or more instances in the training data would be returned (wrongly) 10 times over.
- Standard metrics are based on the confusion matrix formed by predicted and actual codes. For this, we only used the top predicted code based on cosine similarity as opposed to top 10 used for the pilot and Table 1. In the future we will incorporate a parameter which will allow users to control for the number of returned predicted codes for each new entry.

Table 1: Accuracy by data domains based on top 10 predicted codes for each entry from the entire testing dataset

	Pilot Phase			Phase 2		
DATA DOMAIN	**correct predictions**	**total attempted**	**Accuracy**	**correct predictions**	**total attempted**	**Accuracy**
bop6	1,124	1,257	89%	3,603	3,768	96%
1sr	535	536	100%	1,277	1,365	94%
2sr	473	473	100%	1,189	1,264	94%
iip6	408	417	98%	1,116	1,149	97%
bop5	0	62	0%	621	743	84%
dots	91	92	99%	706	706	100%
fs1	142	154	92%	440	524	84%
fs2	97	97	100%	446	499	89%
fsd	67	68	99%	352	355	99%
nag	0	14	0%	285	331	86%
4sr	11	54	20%	276	279	99%
met	0	9	0%	144	276	52%
fas	178	186	96%	250	274	91%
cpi	40	53	75%	194	221	88%
Other common	136	333	41%	536	720	74%
Other non-common	-	-	-	432	491	88%
Total	**3,302**	**3,805**	**87%**	**11,867**	**12,965**	**92%**

Table 2: A sample of classes (out of 5,979 total classes) performance metrics based on top predicted code for each out of sample entry from unique pairs of text and code (label) in training dataset

cts_code	**Recall**	**precision**	**f1**
Class: txg	0.963	0.943	0.953
Class: bfocdaonf_s_bp6	1.000	1.000	1.000
Class: dumu	0.571	0.333	0.421
Class: fodaif_nc	0.900	1.000	0.947
Class: fs_hh_a	1.000	1.000	1.000
Class: iapema_bp6	1.000	1.000	1.000
Class: ggrs_g01	1.000	1.000	1.000
Class: racfampa	1.000	1.000	1.000
Class: ilopcma_bp6	1.000	1.000	1.000
Class: iaocddcip_bp6	1.000	1.000	1.000
Class: falfdf_fx	0.667	1.000	0.800
Class: bfdleis_bp6	1.000	1.000	1.000
Class: fsgdlc	1.000	1.000	1.000
Class: pppi	0.949	0.474	0.632
Class: befde_bp6	1.000	1.000	1.000
Class: nfi	0.667	0.600	0.632
Class: faldomo_nc	1.000	1.000	1.000
Class: pcpiec	0.333	1.000	0.500
Class: fofacflo_fx	1.000	1.000	1.000
…	…	…	…

Table 3: Average performance metrics from Table 2

average_recall	92%
average_precision	89%
average_f1	87%
overall accuracy	72%

4. The R package

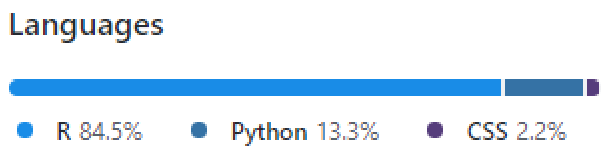
In order to move to a production ready solution, the different programs had to be gathered within a same entity and define these as functions of a same package taking into account all dependencies. With the possibility of running Python code using R, and with the expertise of some team members in building R packages, it was decided to have the end-to-end process in an R package.

The package represents a collection of functions to extract data from country files, process them and codify descriptors following the CTS coding logic based on NLP techniques. It allows the user to process, clean and prepare data from country submission files as well as train the model to classify the extracted time series descriptors and predict CTS codes. The package also includes the code for the Shiny app and a function to launch the Shiny app from the package.

Package details

- **Languages**: the team used different programing languages to create the different functionalities of the tool. Python is used to extract the data from the country file (fairly structured Excel files). R is used to build the rest of the pipeline and the Shiny app. CSS is used to define how HTML elements should be displayed and styled in the Shiny app.

Figure 1: Percentage covered by programming language



- **Users' profiles**: the package was built having two types of users' profiles in mind.
  - **Developer**: This would be the technical user interested in the backend and how the processing and cleaning is done as well as the intricacies of the end-to-end process and how the ML model functions.
  - **Coder**: This represents users interested in running the interface, developed as a shiny app and use it to upload new country submission files, run the predictions and download a version of the uploaded file with predicted CTS codes.
- **Package Structure:** the package encapsulates the different steps the data would go through in our final solution to predict codes. The main functions and a description of these are listed in Table 4

Table 4: List of main functions included in the R package

Function	Description
cts_processdata.R	Extract the data from the country submission files and process them in order to be ready to be used in the next steps of the pipeline. The first part of this process is run in Python. *cts_processdata* is called by the Shiny app as well to process the file uploaded by the user.
cts_train.R	Train the machine learning model to codify the extracted and processed time series descriptors and assign a CTS code. In this case the data also include the actual CTS code assigned to each descriptor in order to do model training.
cts_predict.R	Predict and assign CTS codes to new data using the model trained with *cts_train().* *cts_predict* is called by the Shiny app as well to predict codes the file uploaded by the user.
cts_validate.R	Validate, using a combination of metrics, the results of the predicted codes. *cts_validate* requires as input the predicted codes by the model for each record as well as the actual known CTS codes.
data.R	A collection of data dictionaries exposed to the users of the package and the Shiny app and used throughout the process.
run_ctsapp.R	Launch the Shiny app from the package.
app.R	Shiny app.

The overall pipeline and workflow of the package is illustrated in the figures 2a, 2b and 2c below.

Figure 2a: Data Processing and Model Training

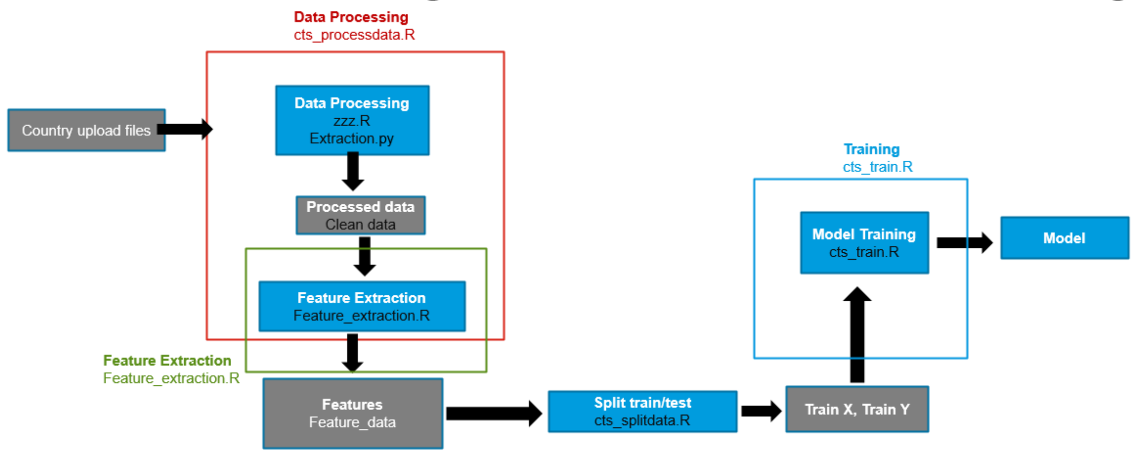


Figure 2b: Prediction and Model Performance

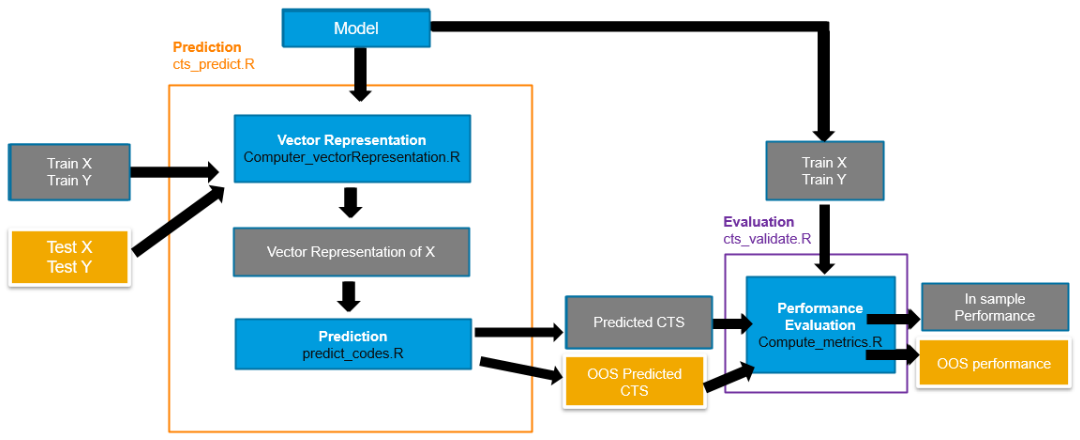
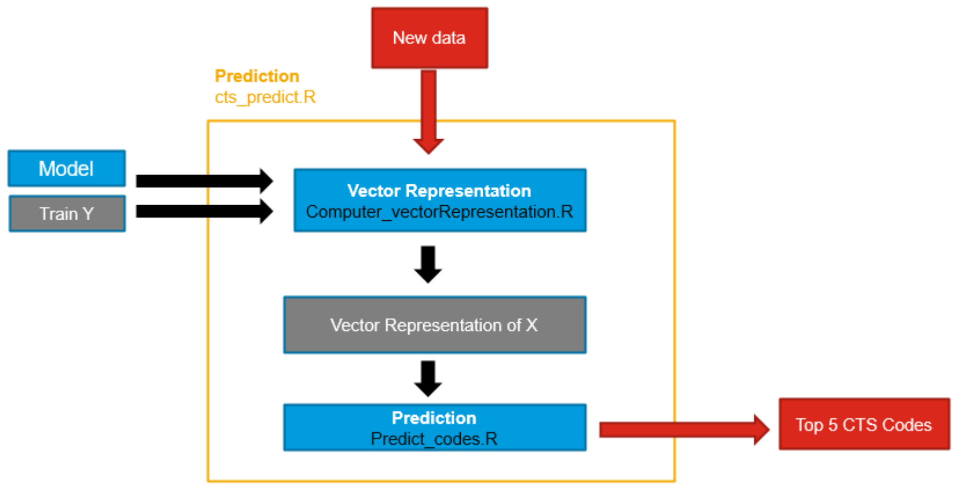


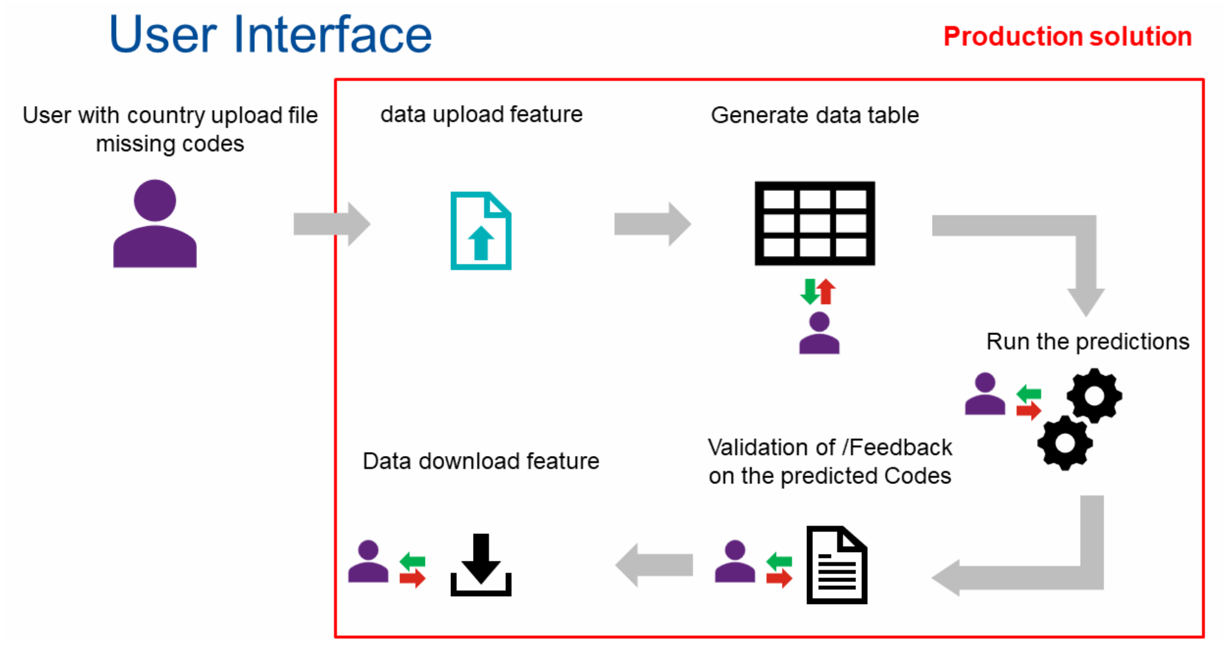
Figure 2c: Prediction on New Data



5. The User Interface

The ultimate goal of this project is the build a tool that can be used by IMF staff working on attributing codes to non-coded indicators from country upload files and help speed up and automate this work. From a user's perspective, having a solution that's easy to use, requires minimal training and is as close as possible to their current process is a key factor in the success of such tool. To facilitate this, the team built an R-Shiny app allowing the user to interact with the different functionalities of the tool and generate predictions following the process shown below.

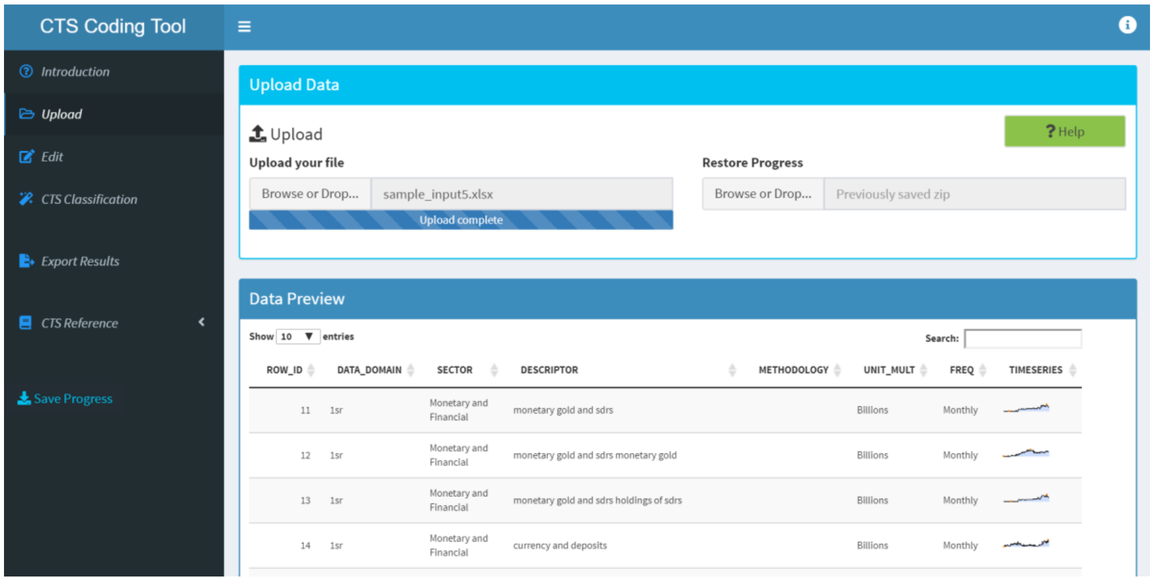
Figure 3: process pipeline when using the Shiny App



The Shiny app was built using the following structure:

- **Introduction**: this is the first page the user sees when running the app and it offers an overview of the application with a user guide or how to run it and a sample file to help understand the needed structure of the upload file and the fields expected by the application.
- **Upload**: This page provides the user with the interface to upload a new file to be coded by the tool following the below steps:
    - Users upload their file into the App
    - The input is processed folowing the same processing steps used to extract and clean the training data used by the model
    - Check the mandatory fields, making sure all mandatory fiels needed by the model are present in the uploaded data
    - Display the data in a table on the app if upload is successful

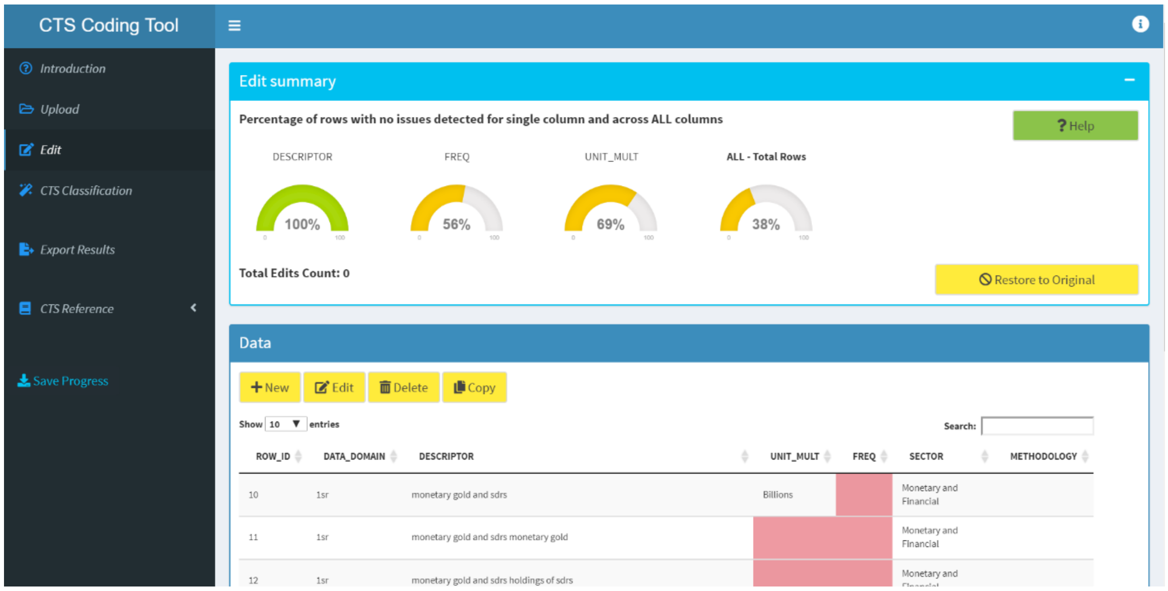
Figure 4: Upload screen of the Shiny App



ROW_ID	DATA_DOMAIN	SECTOR	DESCRIPTOR	METHODOLOGY	UNIT_MULT	FREQ	TIMESERIES
11	1sr	Monetary and Financial	monetary gold and sdrs		Billions	Monthly	
12	1sr	Monetary and Financial	monetary gold and sdrs monetary gold		Billions	Monthly	
13	1sr	Monetary and Financial	monetary gold and sdrs holdings of sdrs		Billions	Monthly	
14	1sr	Monetary and Financial	currency and deposits		Billions	Monthly	

- **Edit**: This page offers to the user the possibilty to adujst the uploaded data before the coding runs. Cells are highlighted in red if an issue is detected: examples include missing values, unrecognized value for variables that can take only a fixed set of values, and so on. The addition of this step aims to potentially maximize the coding outcome by improving and/or adjusting the quality of the input data.

Figure 5: Edit screen of the Shiny App



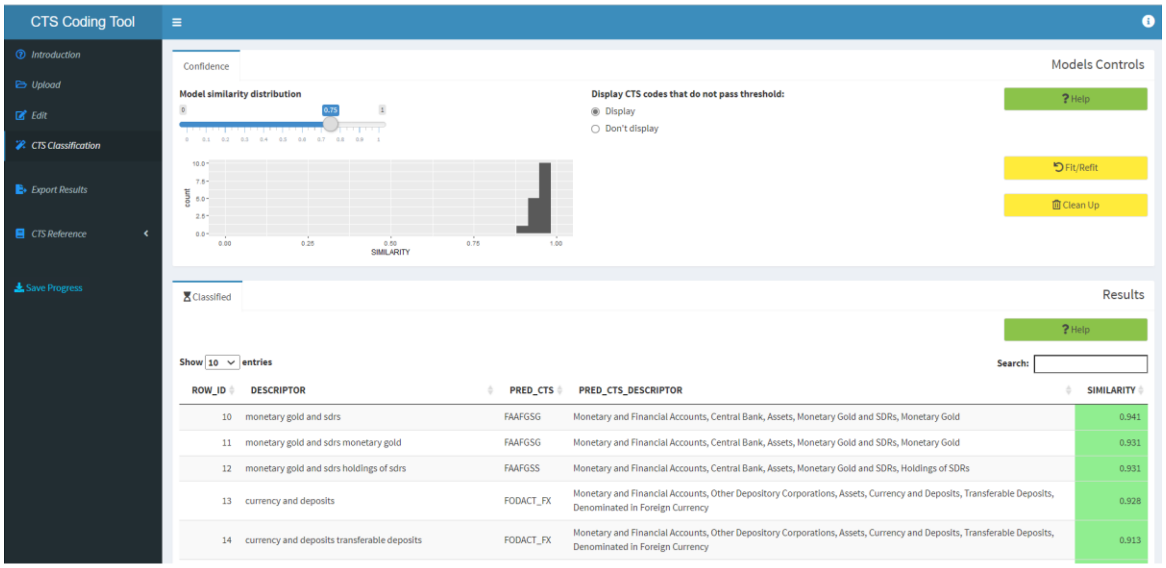
Percentage of rows with no issues detected for single column and across ALL columns

Total Edits Count: 0

ROW_ID	DATA_DOMAIN	DESCRIPTOR	UNIT_MULT	FREQ	SECTOR	METHODOLOGY
10	1sr	monetary gold and sdrs	Billions	🟥	Monetary and Financial	
11	1sr	monetary gold and sdrs monetary gold	🟥	🟥	Monetary and Financial	
12	1sr	monetary gold and sdrs holdings of sdrs	🟥	🟥	Monetary and	

- **Classification**: This page is where the user can run the calssification task and generate codes for the indicators from the uploaded file. Once the coding is run a new column displaying the predicted codes is added to the created table. In addition to the best matched code, the similarity score, assigned by the model to the best matched code and ranging from 0 (lowest) to 1 (highest), is displayed. The color scheme associated with the similarity score is determined by *Model similarity distribution* threshold (green if score is above threshold, yellow if score is around rheshold, red otherwise). This threshold defines a point between 0 ands 1, with default value at 0.75, over which the quality of the assigned code is considered accettable by the user. Assigned codes that don't meet the threshold can also be not displayed in the table if the user prefers so.

- **Figure 6:** CTS Classification screen of the Shiny App



Display CTS codes that do not pass threshold: [✓ Display] Don't display

Show 10 entries

ROW_ID	DESCRIPTOR	PRED_CTS	PRED_CTS_DESCRIPTOR	SIMILARITY
10	monetary gold and sdrs	FAAFGSG	Monetary and Financial Accounts, Central Bank, Assets, Monetary Gold and SDRs, Monetary Gold	0.941
11	monetary gold and sdrs monetary gold	FAAFGSG	Monetary and Financial Accounts, Central Bank, Assets, Monetary Gold and SDRs, Monetary Gold	0.931
12	monetary gold and sdrs holdings of sdrs	FAAFGSS	Monetary and Financial Accounts, Central Bank, Assets, Monetary Gold and SDRs, Holdings of SDRs	0.931
13	currency and deposits	FODACT_FX	Monetary and Financial Accounts, Other Depository Corporations, Assets, Currency and Deposits, Transferable Deposits, Denominated in Foreign Currency	0.928
14	currency and deposits transferable deposits	FODACT_FX	Monetary and Financial Accounts, Other Depository Corporations, Assets, Currency and Deposits, Transferable Deposits, Denominated in Foreign Currency	0.913

- **Export:** In this page, the user can export results of the run predictions. Thay can choose which columns to retain anf the export format of the file.
- **CTS reference:** The CTS reference page offers the user the posibilty to refer to the CTS coding frame directly from the App to search for specific codes or descriptors etc.

6. Project management

The project management aspect was a challenging one in our case, especially having different team members working on a number of competing priorities and assignments. This led us to test different approaches to track the progress of the project as well as distribute the work amongst team members and ensure the delivery of a solution within the set deadline.

- **Azure DevOps:** A first approach was to test Azure DevOps to define the list of tasks and have a clear overview of the pending deliverables and assign these to different team members. Azure DevOps was rather straightforward to use, and allowed creating repositories, sharing and version control the files from the project.

- **GitHub:** Although Azure DevOps provides the possibility to using Git for version control, we decided to shift to GitHub due to the familiarity of most team members. This was a key factor in the build of the R package underlying our production-ready solution. As mentioned earlier in this report, this project had different building blocks that were simultaneously implemented by different team members, and in order to aggregate all the work and build the R package the use of GitHub was key to successfully collaborate and simultaneously work on the different tasks, making sure that all team members had access to the latest versions of the code. In addition to the version control aspect of it, GitHub allowed us to monitor the ongoing work through the issues log feature by creating and assigning new tasks, closing finalized ones and commenting on ongoing work.

- **Weekly Meetings:** The team established a system of weekly meetings to discuss in more detail the project plan, ongoing tasks and pending issues. This was particularly helpful as we had many other high priority tasks outside of this project and having weekly meetings helped us to have a dedicated time for this project. In addition, these weekly meetings were the opportunity to redistribute work when needed based on each team member's workload.

- **The Sprint:** with the deadline approaching, the team felt the need to have dedicated working sessions to wrap up the work for this project. This is an approach we have briefly tested during the first year of the Machine Learning Group and that generated fruitful results. Accordingly, we organized hybrid sprint sessions during the first three weeks of December 2021, both in person and online based on availabilities and preferences of different team members. During these three weeks the team scheduled 4-hour morning sessions dedicated to this project. During these sessions, each team member would work on their specific tasks with the opportunity to have brainstorming sessions, discuss specific issues, get feedback from other team members etc. We truly believe that this helped us a lot to finalize the pending work and improve on some of the already implemented functionalities.

7. Next steps

After the initial development of the R package and Shiny App, the next phase is to share and rollout the tool to a small set of internal users to test the functionalities and receive initial feedback on the overall user experience. This will be done by:

- Sharing the app using an internal R Shiny server
- Organize one-on-one sessions with users to present the app and get direct feedback.

Beside sharing the tool and getting feedback from users for potential improvements, another future task the team would need to tackle is the maintenance and improvement/update of the model. This would be done by:

- Consistently updating the training data with newly coded indicators after review by staff with coding expertise, to ensure the quality of the new added inputs.
- Periodically retrain the model based on the updated training data.
- Looking into improving the overall quality of predictions by updating the model parameters when possible.
- Improving the CTS classification workflow in the Shiny app by allowing the user to review, accept or reject, and overwrite the assigned codes by the model directly in the App.

Annex 1: Domain names table

Domain code	Domain name	CTS sector
BOP	Balance of Payments	External
BOP5	Balance of Payments (BPM5)	External
BOP6	Balance of Payments (BPM6)	External
EXD	External debt	External
EXR	Exchange rates	External
IIP	International Investment Position	External
IIP5	International Investment Position (BPM5)	External
IIP6	International Investment Position (BPM6)	External
ILV1	Official reserve assets	External
ILV2	Template on International Reserves and Foreign Currency Liquidity	External
MET	Merchandise trade	External
DOTS	Direction of Trade Statistics	External
BCG	Budgetary Central Government	Fiscal
CGO	Central government operations	Fiscal
GGO	General Government Operations	Fiscal
BGD	Budgetary Central Government Gross Debt	Fiscal
CGD	Central government debt	Fiscal
GGD	General government gross debt	Fiscal
CBS	Central Bank Survey	Monetary and Financial
ODC	Other Depository Corporations Survey	Monetary and Financial
DCS	Depository Corporations Survey	Monetary and Financial
OFS	Other Financial Corporations Survey	Monetary and Financial
FSI	Financial Soundness Indicators	Monetary and Financial
FAS	Financial Access Survey	Monetary and Financial
INR	Interest rates	Monetary and Financial
MSG	Monetary and Financial Statistics Aggregates	Monetary and Financial
SPI	Share price index	Monetary and Financial
NAG	National Accounts - GDP	Real
CPI	Consumer price indices	Real
LMI	Labor Market Indicators	Real
EMP	Employment	Real
UEM	Unemployment	Real
WOE	Wages/earnings	Real
IND	Industrial Production	Real
POP	Population	Real
PPI	Producer price indices	Real
SOC	Socio-Demographic	Real
X	Not applicable	Not applicable