

TAX GAP ESTIMATION TOOLKIT TRAINING

Programme

Day 1: Monday, 7 October 2024

- 14:00 – 15:00 Opening remarks
Introduction to the Bottom-Up approach with Q&A
- 15:00 – 15:15 Short Break (15 minutes)
- 15:15 – 16:00 Data Preparation theory
- 16:00 – 17:00 Data Preparation practical with Q&A – Group break-out session

Day 2: Tuesday, 8 October 2024

- 14:00 – 15:00 Estimation theory
- 15:00 – 15:15 Short break (15 minutes)
- 15:15 – 16:00 Estimation practical with Q&A – Group break-out session
- 16:00 – 17:00 Interpreting and implementing findings with Q&A
Closing words

Tax gap estimation

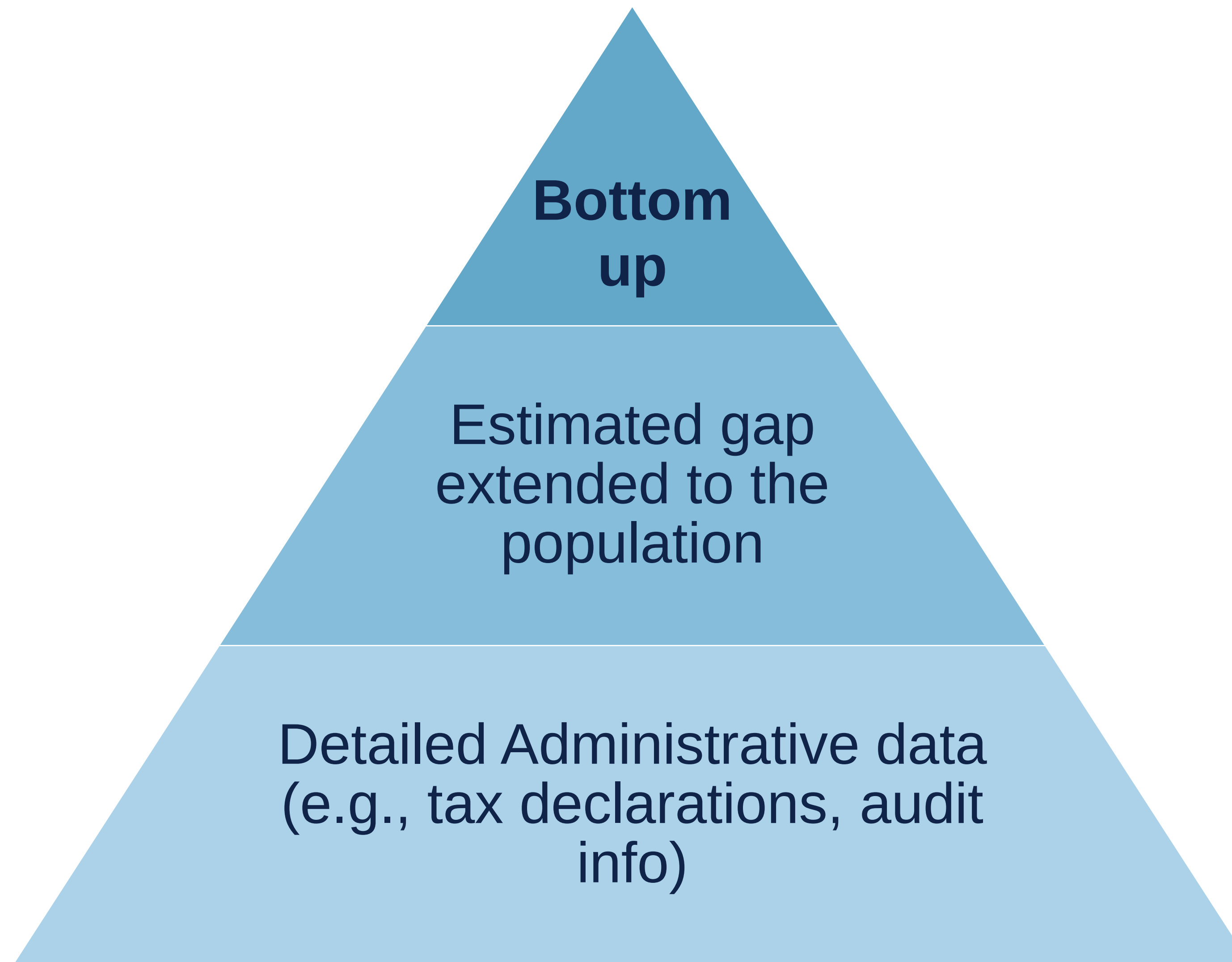


UNU
WIDER

Dr Kwabena Adu-Ababio
Research Associate, UNU-WIDER

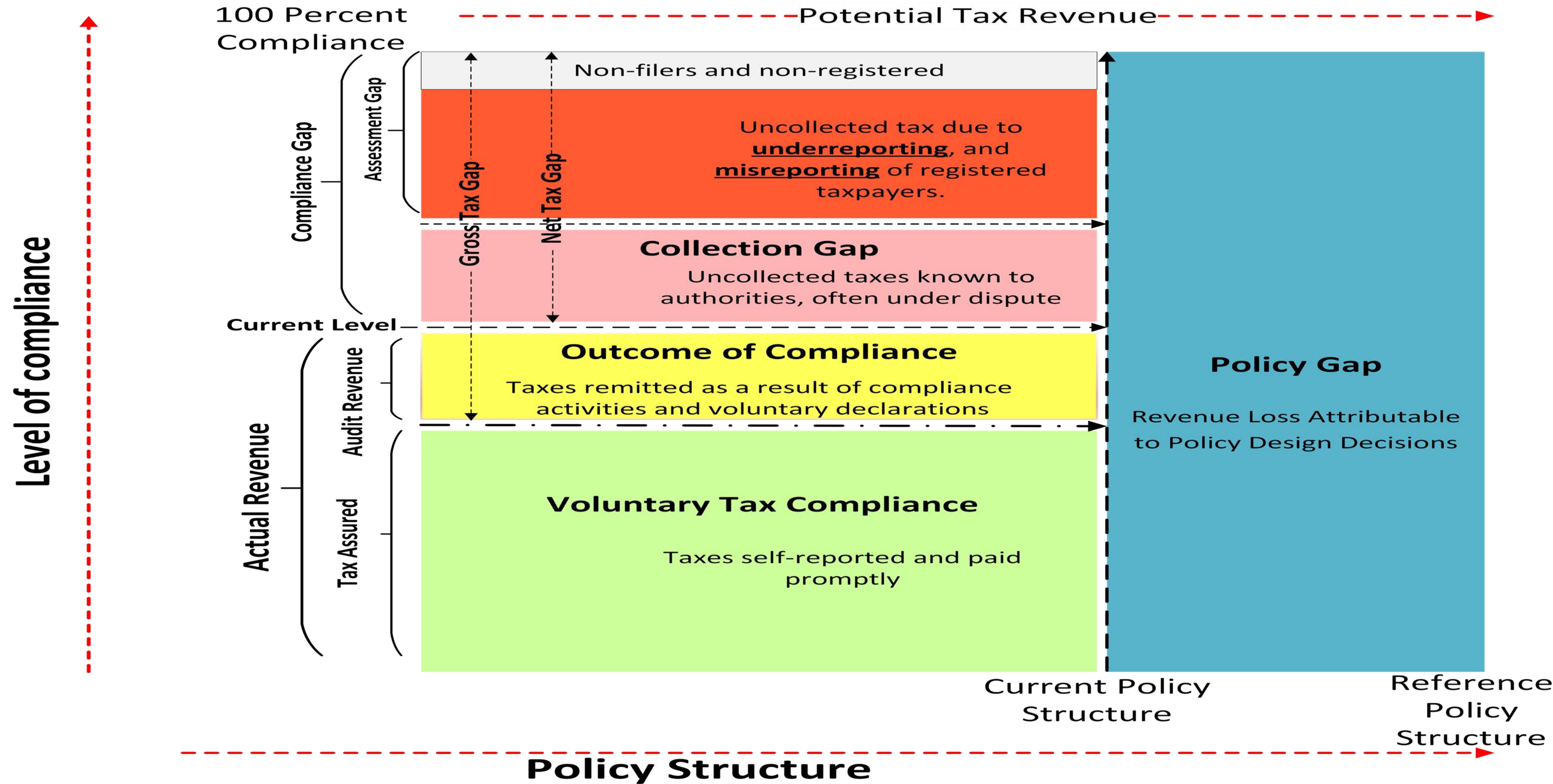
2024 ATI Tax Gap Estimation Toolkit Training | 8 October 2024

Recap



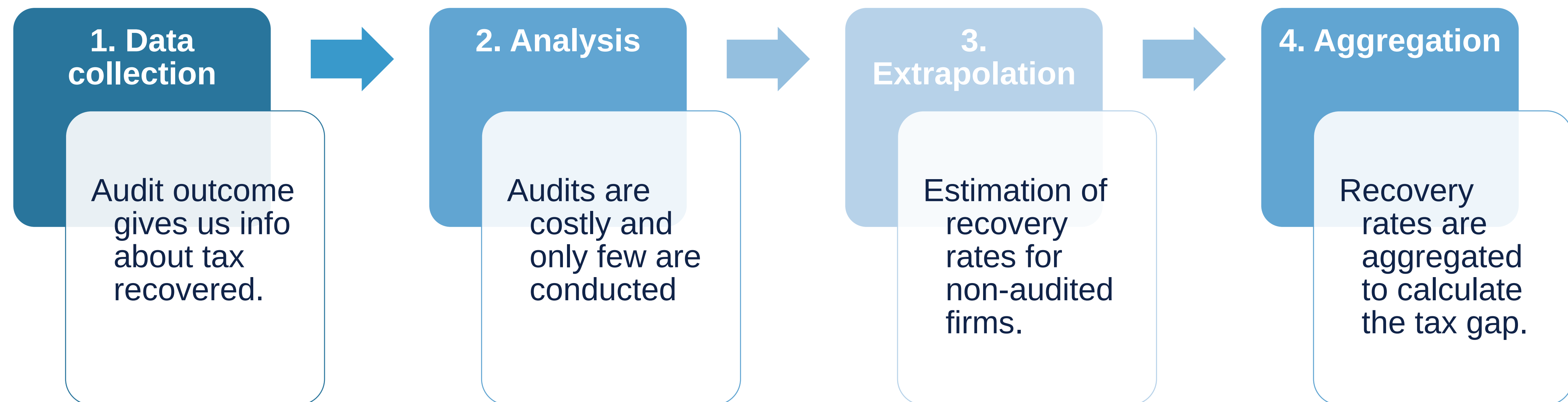
Source: [Australian Tax Office](#)

Recap



Recap of data preparation and tax gap estimation steps

- We have a merged Tax return-Audit data .
- This data will now be used in tax gap estimation.
- Because not all firms are audited, we need to extrapolate and aggregate.



Part 2 of the Toolkit: Tax Gap Estimation using Machine Learning

Extrapolation: Machine learning

- **Machine learning** uses both audited and unaudited taxpayer data to identify patterns and learn the most relevant variables, then predicts the tax potential revenue.

Strengths of Machine Learning:

- Captures complex, non-linear relationships without strict distributional assumptions, and adapts to complex data structures.
- Outperforms traditional statistical methods in predicting tax evasion and can provide more reliable and representative tax gap estimates.
- Addresses selection bias from observable factors by including all known variables influencing audit selection.
- Advanced ML techniques can incorporate methods, like the Heckman correction, to account for unobserved factors influencing selection bias.

Extrapolation: Machine learning

Machine learning algorithms are many → Why use Random Forests for Tax Gap Analysis?

- Performs well even when predictors outnumber observations.
- Effectively manages outliers and atypical values, enhancing prediction accuracy.
- Reduces the risk of overfitting by not using all variables simultaneously.
- Simple to understand, quick to implement, and efficient in practice.

Note: For a detailed explanation of random forest review the [technical note Appendix](#).

Extrapolation: Machine learning

TPIN	Audit Status	Recovery Actual
108469	Audited	17,158.95
105476	Audited	14,921.44
876560	Audited	14,118.49
858323	Audited	16,138.63
684517	Unaudited	
124278	Unaudited	
368696	Unaudited	
335143	Unaudited	
368696	Unaudited	
212796	Unaudited	
166410	Unaudited	

Extrapolation: Machine learning

TPIN	Audit Status	Recovery Actual	Recovery Potential
108469	Audited	17,158.95	
105476	Audited	14,921.44	
876560	Audited	14,118.49	
858323	Audited	16,138.63	
684517	Unaudited		
124278	Unaudited		
368696	Unaudited		
335143	Unaudited		
368696	Unaudited		
212796	Unaudited		
166410	Unaudited		

Extrapolation: Machine learning

TPIN	Audit Status	Recovery Actual	Recovery Potential
108469	Audited	17,158.95	
105476	Audited	14,921.44	
876560	Audited	14,118.49	
858323	Audited	16,138.63	
684517	Unaudited		6,517.63
124278	Unaudited		33,765.42
368696	Unaudited		22,256.54
335143	Unaudited		36,984.14
368696	Unaudited		16,175.93
212796	Unaudited		23,144.84
166410	Unaudited		13,103.30

Tax gap estimation step by step

Readme file

README and Guidance for
Bottom-Up Tax Gap Analysis
Toolkit

1 Overview

2 First Stage: Stata Environment Setup

3 Second Stage: Choosing the Model of
Machine Learning

4 Third Stage: Testing the Model

5 Fourth Stage: Estimating the Tax Gap



README and Guidance for Bottom-Up Tax
Gap Analysis Toolkit

Part 2: Machine Learning

1 Overview

This toolkit, developed by UNU-WIDER/ATI, provides a comprehensive framework for conducting bottom-up tax gap analyses. It utilizes machine learning algorithms to estimate Value Added Tax (VAT), Personal Income Tax (PIT), and Corporate Income Tax (CIT) gaps, specifically designed for Revenue Authorities.

The toolkit consists of two code files: the first part is data management, which is used to clean the tax return and audit data and merge them, so it is ready for machine learning analysis using the second code file.

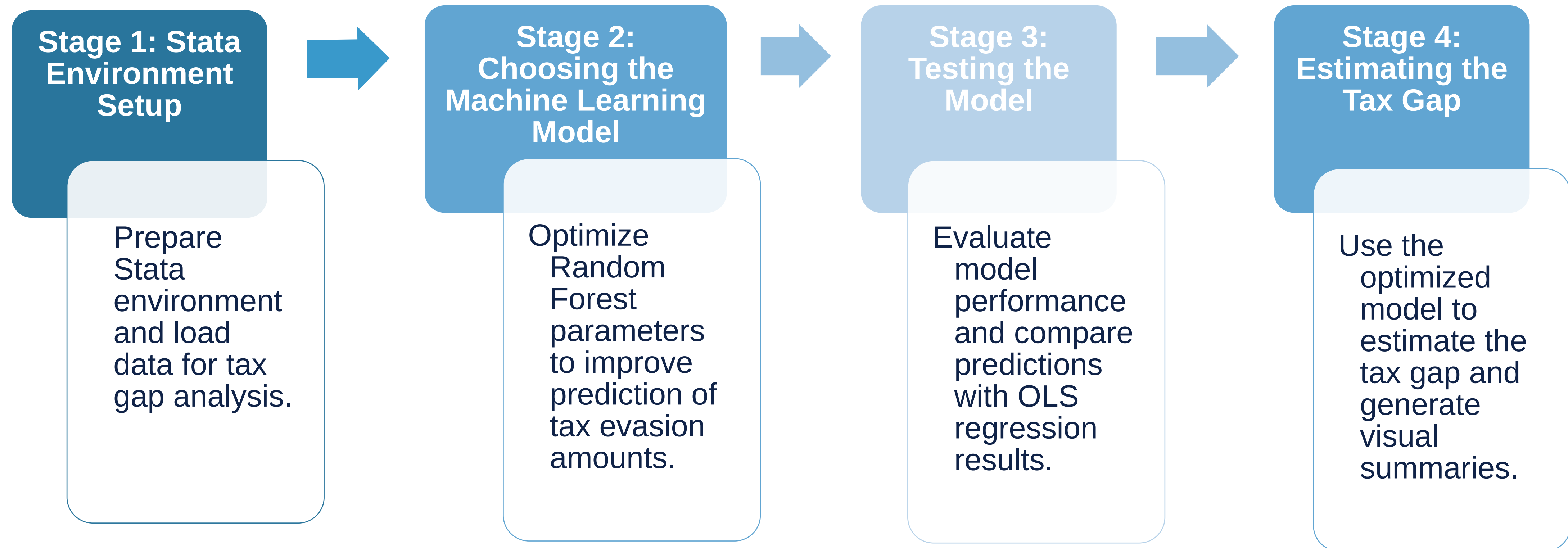
Given the steps and recommendations in the data data management file, it is important to note that this document assumes the structure of the data advised there. This means we have different databases for each tax type (VAT, CIT, or PIT). In case you cannot differentiate between those tax types, the process is still valid but assumes the variable to predict the general misreporting amount and makes some assumptions to obtain a disaggregated estimation for each tax type. We will indicate what procedure you must follow if you have an aggregate measure of misreporting instead of each tax type when applicable.

Version:
Date: 28/06/2024

Prerequisites:
To use this toolkit, we must have [Stata software](#) installed on your computer. Additionally, the data requirements depend on the tax gap you want to analyze. Whether it is CIT, VAT, or PIT, the data should be cleaned and merged with audit data following the Data Management Part 1 of the toolkit.

Overview of Machine Learning Tax Gap Estimation Stages

The tax gap estimation part of the toolkit consists of **four main stages**:



Stage 1: Stata Environment Setup

Install Random Forest package

```
ssc install rforest
```

Prepare Stata environment

```
set more off  
clear matrix  
set linesize 200  
set matsize 800  
cap log close  
version 14 /* Use Stata Version 14 */
```

Set up global directory paths

```
global dir "C:\Users\<YOUR PATH>\"  
  
global data "$dir\Data"  
global graphs "$dir\Graphs"  
global xlsout "$dir\Excel"  
global dopath "$dir\Dofile"  
global results "$dir\Results"  
global log "$dir\Log"
```

Import data

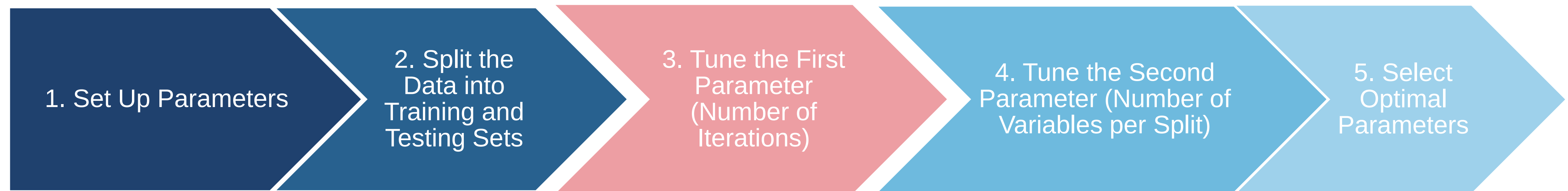
```
64 /// 4. Importing Data  
65  
66 use "$data/ZZ" // Insert the database to use  
67
```

Stage 2: Choosing the Machine Learning Model

Requirements:

- **Cleaned Dataset:** Ensure your dataset is cleaned and merged as per Part 1 of the toolkit.
- **Variables Identification:**
 - **Audited Regions Variable:** A variable indicating regions where audits were conducted.
 - **Evaded Tax Variable:** A variable representing the amount of tax evaded (e.g., VAT, CIT, PIT).
- **Relevant Variables for Prediction:** All variables that might help predict tax evasion.

Steps:



Stage 2: Choosing the Machine Learning Model

1. Set Up Parameters:

Identify Audited Regions:

```
generate reg_audit = (region == YY | region == TT)
```

Create Evaded Tax Variable:

```
generate evaded = XX
```

Restrict Data to Audited Periods in Audited Regions:

```
keep if evaded !=. & reg_audit==1
```

Stage 2: Choosing the Machine Learning Model

1. Set Up Parameters:

Identify Audited Regions:

```
generate reg_audit = (region == YY | region == TT)
```

Create Evaded Tax Variable:

```
generate evaded = XX
```

Restrict Data to Audited Periods in Audited Regions:

```
keep if evaded !=. & reg_audit==1
```

Stage 2: Choosing the Machine Learning Model

2. Split the Data into Training and Testing Sets:

1. **Training sample:** This is the part of the sample where we run the model. In this sense, we are training the model on this subsample.
 2. **Testing sample:** In this part, we will compare the predictions from the model with the actual value of the variable. In this sense, we will test our predictions on this subsample.
- The testing sample should be a percentage of the total sample. For instance, you could split the sample into 50% for testing and 50% for training. Remember to replace " $nn + 1 / NN$ " with numbers based on the sample size. Hence, if there are 100 observations, nn would be 50 and NN would be 100.

Set Seed for
Reproducibility:

```
set seed 201807
```

Create a Random
Variable for Splitting:

```
92 generate u = uniform() // We create a variable based on a uniform distribution to divide the sample.  
93 sort u, stable // Sorting the indicator variable.
```

Stage 2: Choosing the Machine Learning Model

3. Tune the First Parameter (Number of Iterations):

Initialize Auxiliary Variables:

```
118 generate out_of_bag_error1 = . // Create an auxiliary variable to capture the error in the training sample.
119 generate validation_error = . // Create an auxiliary variable to capture the error in the testing sample.
120 generate iter1 = . // Creating auxiliary variables to produce the iterations to choose the critical parameters.
121 local j = 0 // Creating auxiliary variable.
122
123 scalar t1 = c(current_time) // This is to measure the time in this step.
```

Run Loop to Find Optimal Iterations:

```
124
125 forvalues i = 10(5)500{
126     local j = `j' + 1
127     rforest evaded /VARIABLES/ in 1/nn, type(reg) iter(`i') numvars(1)
128     quietly replace iter = `i' in `j'
129     quietly replace out_of_bag_error1 = `e(OOB_Error)' in `j'
130     predict pred in nn+1/NN
131     quietly replace validation_error = `e(RMSE)' in `j'
132     drop pred
133 }
134
135 scalar t2 = c(current_time)
136
137 display (clock(t2, "hms") - clock(t1, "hms")) / 1000 " seconds" // This displays the time spent in the first estimation.
138
139 label variable out_of_bag_error1 "Out-of-bag error"
140 label variable iter1 "Iterations"
141 label variable validation_error "Validation RMSE"
```

Plot Errors to Identify Optimal Iterations:

```
146 graph twoway (scatter out_of_bag_error1 iter1, mcolor(blue) msize(tiny)) (scatter validation_error iter1, mcolor(red) msize(tiny))
147 graph save Graph "$graphs/ml_iter.gph", replace
148 graph export "$graphs/ml_iter.pdf", as(pdf) replace
```

Stage 2: Choosing the Machine Learning Model

- 4. Tune the Second Parameter (Number of Variables per Split):

Initialize
Auxiliary
Variables:

Identify Optimal
Number of
Variables:

Identify Optimal
Number of
Variables:

Plot Errors to
Confirm Optimal
Number of
Variables:

```
165 generate oob_error = . // Create an auxiliary variable to capture the error in the training sample.
166 generate nvars = . // Create an auxiliary variable for the number of variables that are chosen in each split.
167 generate val_error = . // Create an auxiliary variable to capture the error in the testing sample.
168 local j = 0
169
170 capture frame drop mydata2
171 scalar t3 = c(current_time)
```

```
172
173 forvalues i = 1(1)VV{
174     local j = `j' + 1
175     rforest evaded /VARIABLES/ in 1/nn, type(reg) iter(300) numvars(`i')
176     quietly replace nvars = `i' in `j'
177     quietly replace oob_error = `e(OOB_Error)' in `j'
178     predict pre in nn+1/NN
179     quietly replace val_error = `e(RMSE)' in `j'
180     drop pre
181 }
182
183 scalar t4 = c(current_time)
184
185 display (clock(t4, "hms") - clock(t3, "hms")) / 1000 " seconds"
186
```

```
194
195 frame put val_error nvars, into(mydata2)
196
197 frame mydata2{
198     sort val_error, stable
199     local min_val_err = val_error[1]
200     local min_nvars = nvars[1]
201 }
202
203 frame drop mydata2
204 display "Minimum Error: `min_val_err'; Corresponding number of variables `min_nvars'" // This number is the second relevant parameter.
205
```

```
207 graph twoway (scatter oob_error nvars, mcolor(blue) msize(tiny)) (scatter val_error nvars, mcolor(red) msize(tiny))
208 graph save Graph "$graphs/ml_var.gph", replace
209 graph export "$graphs/ml_var.pdf", as(pdf) replace
210
```

Stage 2: Choosing the Machine Learning Model

5. Select Optimal Parameters:

Use the optimal number of iterations and variables per split identified in the previous steps for the final model.

Stata/MP 16.0 - C:\Users\sebcas\OneDrive - University of Helsinki\FIT\WIDER Tax Gap\Codes\Exercise\WAT\total_vat.dta

File Edit Data Graphics Statistics User Window Help

History

Filter commands here

Command

1 doedit "C:\Users\sebcas\O...

2 use "C:\Users\sebcas\One...

3 help collect

4 collect clear

5 sum totalvatdue

6 bro

7 clear

8 use "C:\Users\sebcas\One...

9 sum vat_evasion

10 collect get row 1 = "e(mean)... 7

11 collect get row1 = "e(mean)...

12 collect get rj(mean)

13 collect get rj(var) rj(max) rj(m...

14 collect get row1 = "rj(mean)..."

15 collect get row1 = "e(sdi)... 198

16 collect get row2 = "r(sdi)... ta...

17 sum vat_payment

18 collect get row1 = "rj(mean)..."

19 collect get row2 = "r(sdi)... ta...

20 sum vat_taxunregistered_pu...

21 collect get row1 = "rj(mean)..."

22 collect get row2 = "r(sdi)... ta...

23 sum total_vat_sales

24 collect get row1 = "rj(mean)..."

25 collect get row2 = "r(sdi)... ta...

26 collect layout (result) (col)

27 collect layout (result) (col co...

28 collect label dim col1 "Traini...

29 collect label dim col2 "Test"

30 collect style header col1, titl...

31 collect style header col2, titl...

32 collect style column, dups(c...

33 collect style cell cell_type(co...

34 collect style cell col#cell_ty...

35 collect style cell result1, bor...

36 collect title "Table. Summary...

37 collect style title, smd(input)

38 collect label levels result1 r... 198

39 row1 "RMSE" 199

40 row2 "R-square" 199

41 199

42 collect preview

43 sum vat_evasion

44 collect get row1 = "rj(mean)..."

45 collect get row2 = "r(sdi)... ta...

46 sum vat_payment

47 collect get row1 = "rj(mean)..."

48 collect get row2 = "r(sdi)... ta...

49 collect get row2 = "r(sdi)... ta...

50 sum vat_taxunregistered_pu...

51 collect get row1 = "rj(mean)..."

52 collect get row2 = "r(sdi)... ta...

53 sum total_vat_sales

54 collect get row1 = "rj(mean)..."

55 collect get row2 = "r(sdi)... ta...

56 rln ~C:\Users\sebcas\OneDrive - University of Helsinki\FIT\WIDER Tax Gap\Codes

. display (clock(t4, "hms") - clock(t3, "hms")) / 1000 " seconds"

7207 seconds

.
. .
. .
. label variable oob_error "Out-of-bag error"

. label variable val_error "Validation RMSE"

. label variable nvars "Number of variables randomly selected at each split"

.
. // In the following plot, we see the out-of-bag error and the validation error. The critical number is chosen by looking at the iteration that
> minimizes the error in the testing sample.
. // For example, if at iteration 20, the minimum error is obtained, you need to choose 20 as the critical number. Before plotting, we will intr
> oduce a code that shows you the number of iterations that minimize the error and the value of the error.
. .
. frame put val_error nvars, into(mydata2)
. .
. frame mydata2{
. . sort val_error, stable
. . local min_val_err = val_error[1]
. . local min_nvars = nvars[1]
. . }
. .
. frame drop mydata2
. .
. display "Minimum Error: `min\_val\_err'; Corresponding number of variables `min_nvars'" // This number is the second relevant parameter.
Minimum Error: 409.18212890625; Corresponding number of variables 16
. .
. .
. graph twoway (scatter oob_error nvars, mcolor(blue) msize(tiny)) (scatter val_error nvars, mcolor(red) msize(tiny))
. .
. graph save Graph "\$graphs/ml_var.gph", replace
(file C:\Users\sebcas\OneDrive - University of Helsinki\FIT\WIDER Tax Gap\Codes\Testing\Graphs\ml_var.gph not found)
file C:\Users\sebcas\OneDrive - University of Helsinki\FIT\WIDER Tax Gap\Codes\Testing\Graphs\ml_var.gph saved
. .
. graph export "\$graphs/ml_var.pdf", as(pdf) replace
file C:\Users\sebcas\OneDrive - University of Helsinki\FIT\WIDER Tax Gap\Codes\Testing\Graphs\ml_var.pdf saved as PDF format

Variables

Filter variables here

Name

Label

tpin

year

month

tax_sales

vat_tax_sales

zero_rate_sales

exempt_sales

import_sales

vat_import_sales

export_sales

capital_sales

nondeduc_sales

taxunregistered_sales

vat_taxunregistered_s...

total_sales

total_vat_sales

tax_purchases

vat_tax_purchases

zero_rate_purchases

exempt_purchases

import_purchases

vat_import_purchases

export_purchases

capital_purchases

Properties

Variables

Name

Label

Type

Format

Value label

Notes

iter1

iterations

float

%8.0g

Data

Frame

Filename

Label

Notes

Variables

Observations

Size

Memory

Sorted by

default

total_vat.dta

47

36,000

6.59M

64M

u



09/10/2024 22

Stage 2: Choosing the Machine Learning Model

5. Select Optimal Parameters:

The following Figure 1 shows the model performance over iterations for the first parameter.

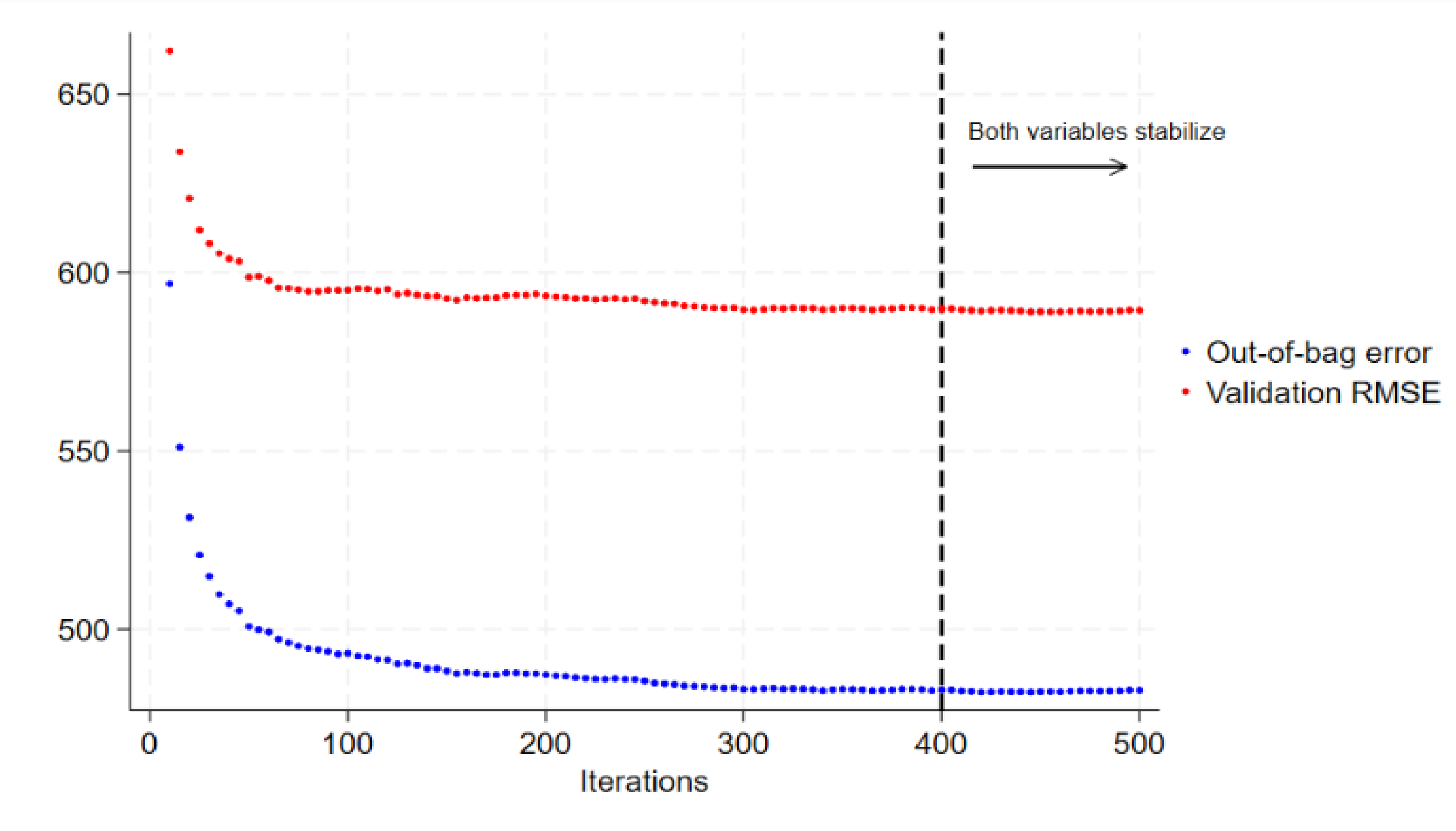


Figure 1: Out-of-bag error and Validation RMSE over Iterations for the first parameter

The following Figure 3 shows the model performance over iterations for the second parameter.

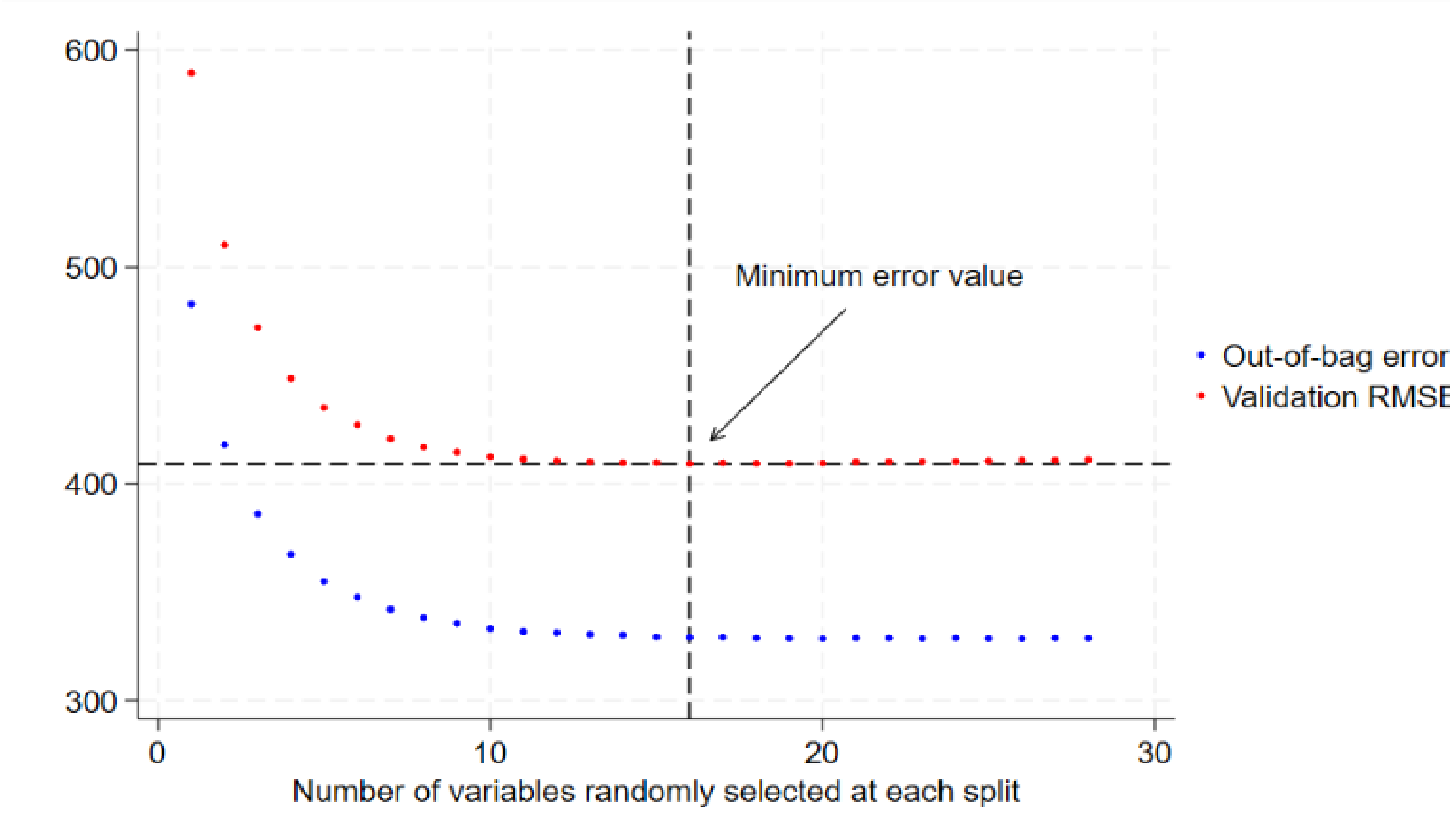


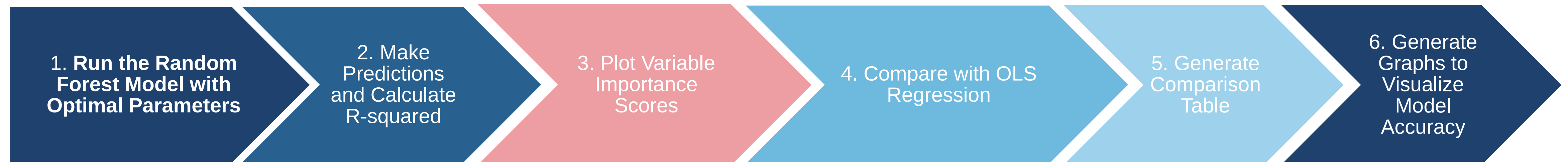
Figure 3: Out-of-bag error and Validation RMSE over Iterations for the second parameter

Stage 3: Testing the Machine Learning Model

Requirements:

- **Optimal Parameters:** From Stage 2 (e.g., Optimal Iterations and Optimal number of Variables).
- **Training and Testing Data:** Split datasets from Stage 2.
- **Predictor Variables:** The same variables used in Stage 2.

Steps:



Stage 3: Testing the Machine Learning Model

1. Run the Random Forest Model with Optimal Parameters:

```
220
221 capture frame drop mydata2
222 scalar t5 = c(current_time)
223
224 rforest evaded /VARIABLES/ in 1/nn, type(reg) iter(300) numvars(20)
225
226 scalar t6 = c(current_time)
227
228 display (clock(t6, "hms") - clock(t5, "hms")) / 1000 " seconds"
229
230
231 ereturn list OOB_Error
232 scalar oob_error = `e(OOB_Error)'
233
```

Stage 3: Testing the Machine Learning Model

2. Make Predictions and Calculate R-squared:

- On training data:

```
235 predict tot_pred    // We predict the values for all the sample
236
237 corr evaded tot_pred in 1/nn
238 scalar r_m1 = `r(rho)`
239 di r(rho)^2 // We obtain the r square for the prediction in the training data
240
```

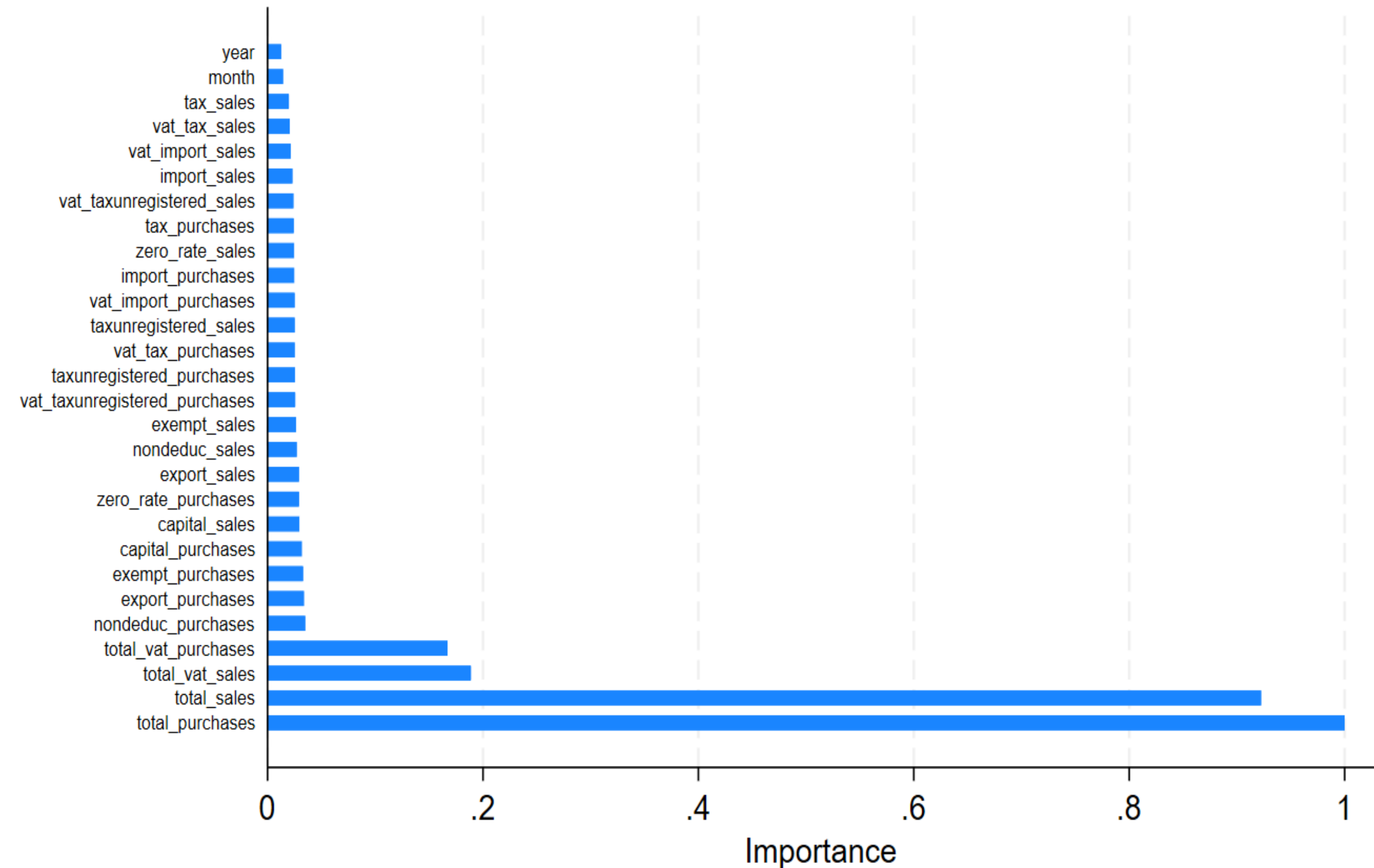
- On testing data:

```
241
242 predict pre in nn+1/NN // We predic thte values only for the testing data
243 ereturn list RMSE
244 scalar rmse_m1 = `e(RMSE)`
245
246
247 corr evaded pre in nn+1/NN
248 scalar r_m12 = `r(rho)`
249 di r(rho)^2 // We obtain the r square for the prediction in the testing data
250
```

Stage 3: Testing the Machine Learning Model

3. Plot Variable Importance Scores:

```
253
254 matrix importance2 = e(importance)
255 svmat importance2
256 generate importid2=""
257
258 local min_importance = ii
259 local mynames: rownames importance2
260 local k: word count `mynames'
261
262 if `k' > _N {
263     set obs `k'
264 }
265
266 forvalues i = 1(1)`k' {
267     local aword: word `i' of `mynames'
268     local alabel: variable label `aword'
269     if ("`alabel'" != "") qui replace importid2 = "`alabel'" in `i'
270     else quietly replace importid2 = "`aword'" in `i'
271 }
272
273 graph hbar (mean) importance2 if importance2 > `min_importance', over(importid2, sort(1)
274     label(labsize(2))) ytitle(Importance) note("Only variables with an importance score larger
275     than `min_importance' % are reported.")
276 graph save Graph "$graphs/ml_var_rel.gph", replace
277 graph export "$graphs/ml_var_rel.pdf", as(pdf) replace
278
279 * With the following command, we save the index for each variable. With this information,
280 you can analyze how relevant are each variable used in your model to predict the non-paid
281 tax.
282
283 putexcel set "$xlsout/ml_var_rel_tot.xlsx", replace
284 putexcel A3 = matrix(importance2), rownames nformat(number_d2)
```



Only variables with an importance score larger than 0.1 % are reported.

Stage 3: Testing the Machine Learning Model

4. Compare with OLS Regression:

- Run OLS Regression:

```
289 quietly reg evaded /VARIABLES/ in 1/nn
290 ereturn list r2 // We obtain the r square to compare with the ML in the training data
291 scalar r_ols = `e(r2)'
292 scalar rmse_ols = `e(rmse)'
293
```

- Predict and Calculate R-squared on Testing Data:

```
298 corr evaded ev in nn+1/NN
299 scalar r_ols2 = `r(rho)'
300 di r(rho)^2 // We obtain the r square to compare with the ML in the training data
301
302
303 generate diff_sqr2= (evaded - ev)^2
304 summarize diff_sqr
305 scalar rmse_ols2 = `r(mean)'
306
307
308 summarize evaded in 1/nn
309 scalar tra_mean = `r(mean)'
310 scalar sd = `r(sd)'
311
312 summarize evaded in nn+1/NN
313 scalar tes_mean = `r(mean)'
314 scalar sd2 = `r(sd)'
315
```

Stage 3: Testing the Machine Learning Model

5. Generate Comparison Table:

Code lines 318-400:
Uses the collect
command to organize
and format a table that
compares the OLS
and ML models based
on R-square and
RMSE metrics.

Table 1: Summary Statistics

Metric	Training ML	Training OLS	Test ML	Test OLS
RMSE	482.95	509.11	409.18	502.15
RMSE/Mean (evasion)	0.450	0.474	0.386	0.474
R-square	0.964	0.772	0.852	0.777
Mean (evasion)	1073.44	1073.44	1059.05	1059.05

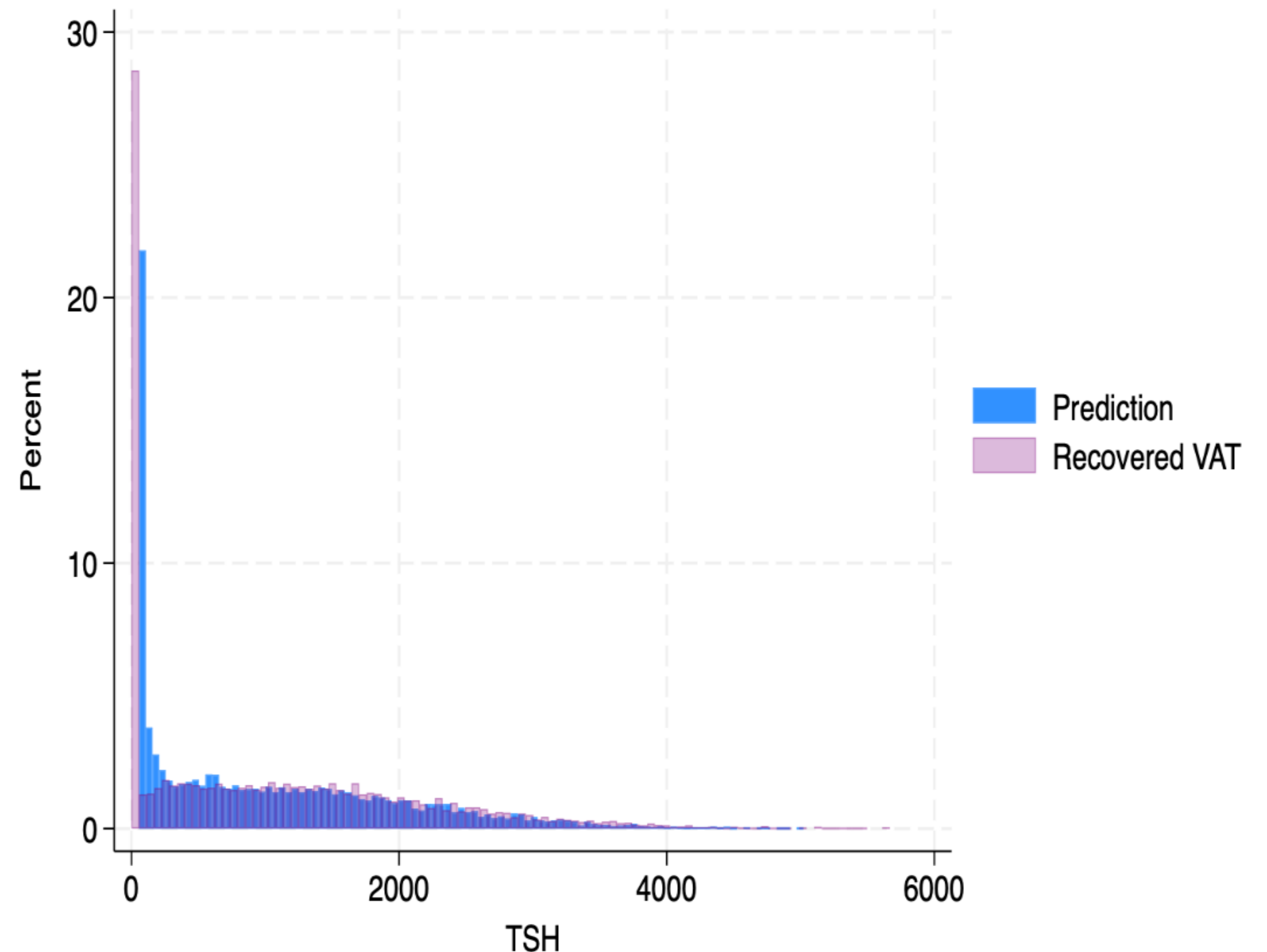
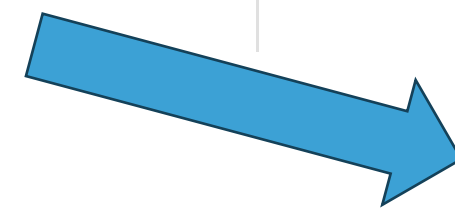
Note: Higher R-square values for the ML model suggest a better fit to the data

Stage 3: Testing the Machine Learning Model

6. Generate Graphs to Visualize Model Accuracy:

Histogram of Predicted vs. Actual Values:

```
406 graph twoway (histogram tot_pred in nn+1/NN, percent bin(100)) (histogram evaded in nn+1/NN  
    5 , percent bin(100) fcolor(purple%30) lcolor(purple%30)), xtitle("`TSH`") legend(order(1  
    5 "Prediction" 2 "Recovered VAT"))  
407 graph save Graph "$graphs/histogram_prediction1.gph", replace  
408 graph export "$graphs/histogram_prediction1.pdf", as(pdf) replace  
409
```

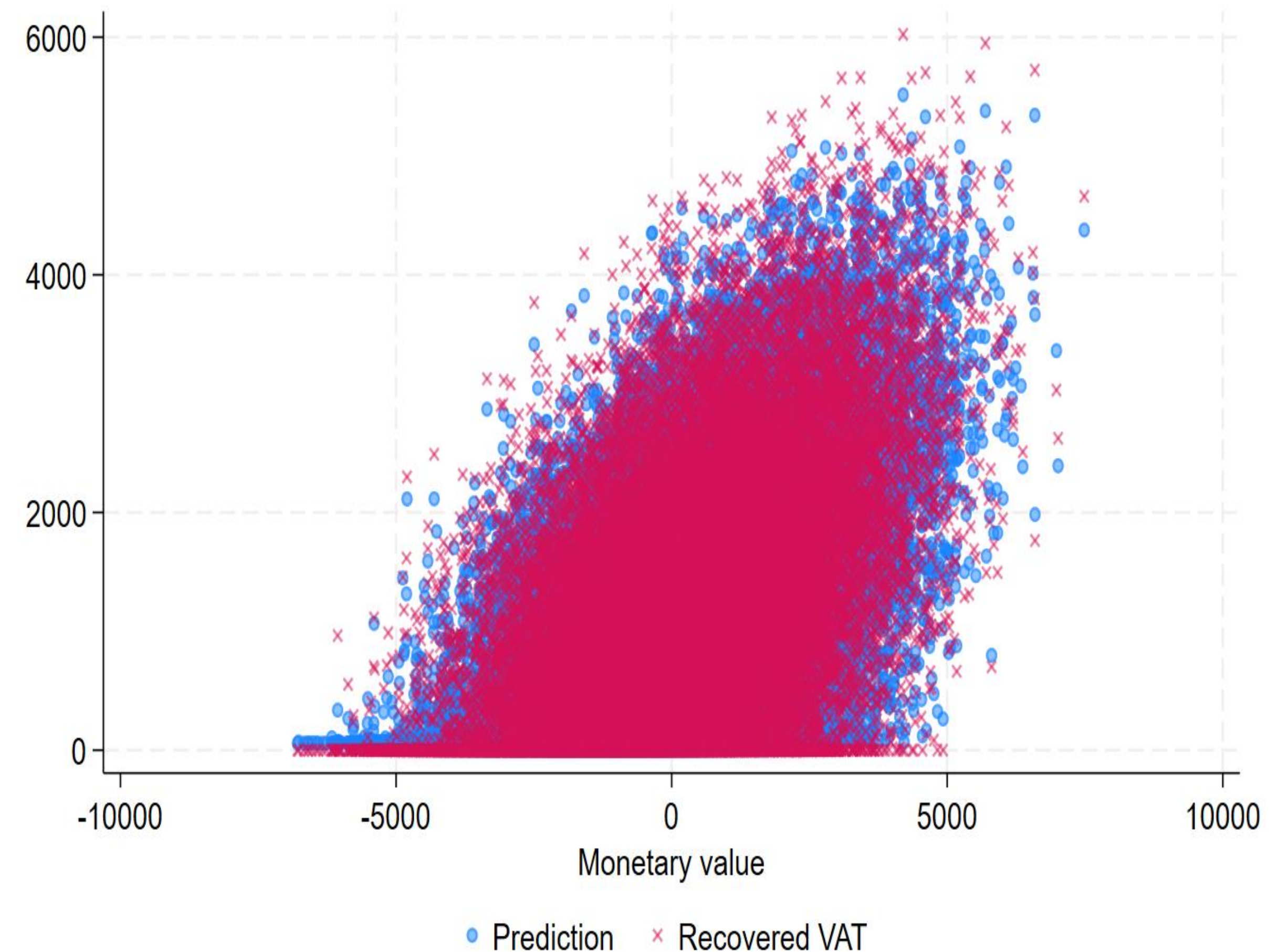
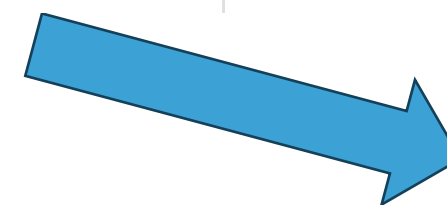


Stage 3: Testing the Machine Learning Model

6. Generate Graphs to Visualize Model Accuracy:

Scatterplot of Predicted vs. Actual Values:

```
413 graph twoway (scatter tot_pred tax_vle, sort mcolor(%50)) (scatter evaded tax_vle, sort  
↳ mcolor(%50) msymbol(lgx)), legend(rows(1) position(6)) xtitle("Monetary Value") legend(  
↳ order(1 "Prediction" 2 "Recovered VAT"))  
414 graph save Graph "$graphs/histogram_prediction2.gph", replace  
415 graph export "$graphs/histogram_prediction2.pdf", as(pdf) replace  
416
```



Stage 4: Estimating the Tax Gap

Requirements:

Full Dataset: Reload the original dataset with all observations.

Optimal Model Parameters: From previous stages.

Variables for Grouping: Variables to group by (e.g., year, industry, region).

Steps:



Stage 4: Estimating the Tax Gap

1. Prepare Data for Prediction:

Clear and Reload
Data:

```
428  
429 clear  
430  
431 use "$data/ZZ" // Insert the database to use
```

Generate Necessary
Variables:

```
433 generate reg_audit = YY // Insert YY as the indicator variable for the regions  
434      (geographical or tax administrative) where auditing was conducted.  
435 generate evaded = XX // Insert XX as the variable that compasses the taxes that  
436      taxpayers do not pay to estimate the gap (VAT, CIT, or PIT). Generally, this is found by  
437      auditing. This variable cannot have missing values.  
438 generate indicator = (evaded !=. & reg_audit==1)  
439 generate mimic_tax = evaded  
440 replace mimic_tax= 0 if evaded==.  
441
```

Stage 4: Estimating the Tax Gap

2. Run the Random Forest Model on Full Dataset:

Predict Unpaid Tax:

```
443  
444 rforest mimic_tax /VARIABLES/ if indicator==1, type(reg) iter(II) numvars(VV)  
445 predict assessment if indicator==0 & reg_audit==1  
446 replace assessment = evaded if indicator==1 & reg_audit==1  
447
```

Handle Negative Predictions (if applicable):

```
451 sum assessment, d  
452  
453 // If your variable evaded contains negative values, follow this step. If not, you can  
454    ⚡ omit it.  
455 generate assessment_2 = assessment  
456 replace assessment_2 = 0 if assessment_2<0 // We impute that negative values are actually  
457    ⚡ zero, by construction.  
457
```

Stage 4: Estimating the Tax Gap

3. Calculate the Tax Gap:

Option A: If No Negative Values in 'evaded' Variable

Calculate Tax Gap:

```
477 preserve
478
479 collapse (sum) assessment tax if reg_audit==1, by(year)
480
481 generate v01 = assessment+abs(tax)
482
483 generate vat_gap01 = (assessment/v01)*100
484
```

Plot the Tax Gap Percentage:

```
486 graph bar (mean) vat_gap01, over(year) blabel(bar, size(vsmall) format(%9.2f)) legend(
↳ label(1 "TAX") ) ytitle(% of Evasion) title("Percentage of tax evaded") subtitle("by
↳ audited tax regions") note("Aggregate values.")
487 graph save Graph "$graphs/vat_gap_year.gph", replace
488 graph export "$graphs/vat_gap_year.pdf", as(pdf) replace
```

Export Summary Statistics:

```
490 estpost tabstat vat_gap01 , by(year) stat(max)col(stat)
491 esttab . using "$xlsout/results.csv", cells("max(lab(Max) fmt(%15.2fc))") noobs
↳ nonumber eqlabels(`e(labels)') varwidth(20) replace
492
493
494 restore
495
496
```

Stage 4: Estimating the Tax Gap

3. Calculate the Tax Gap:

Option B: If 'evaded' Variable Contains Negative Values

Calculate Tax Gap:

```
520 preserve
521
522 collapse (sum) assessment_2 tax if reg_audit==1, by(var_analysis)
523
524 generate v01 = assessment_2+abs(tax)
525
526 generate vat_gap01 = (assessment_2/v01)*100
527
```

Plot the Tax Gap Percentage:

```
529 graph bar (mean) vat_gap01, over(var_analysis) blabel(bar, size(vsmall) format(%9.2f
530 )) legend( label(1 "TAX") ) ytitle(% of Evasion) title("Percentage of tax evaded")
531 subtitle("by audited tax regions") note("Agregate values.")
532 graph save Graph "$graphs/vat_gap.gph", replace
533 graph export "$graphs/vat_gap.pdf", as(pdf) replace
534
```

Export Summary Statistics:

```
532
533 estpost tabstat vat_gap01 , by(var_analysis) stat(max)col(stat)
534 esttab . using "$xlsout/results.csv", cells("max(lab(Max) fmt(%15.2fc))") noobs
535 nonumber eqlabels(`e(labels)') varwidth(20) append
536
537 restore
```

Stage 4: Estimating the Tax Gap

4. Obtain Nominal Values for Potential Revenue, Total Tax Collected, and Potential Revenue Lost

A. Overall Results by Year

Collapse Data and Calculate Potential Revenue:

```
543 preserve
544
545 collapse (sum) assessment tax if reg_audit==1, by(year)
546
547 generate potential_revenue = assessment+abs(tax)
548
```

Plot Revenue Consequences:

```
550 graph bar (mean) potential_revenue assessment tax, over(year) blabel(bar, size(vsmall
↳ ) format(%9.2f)) legend( label(1 "Potential Revenue") label(2 "Revenue Lost
↳ (predicted)") label(3 "Revenue Collected") ) ytitle(Monetary Unit) title("Revenue
↳ Consequences") subtitle("by audited tax regions") note("Aggregate values.")
551 graph save Graph "$graphs/revenue_gap.gph", replace
552 graph export "$graphs/revenue_gap.pdf", as(pdf) replace
553
```

Export Summary Statistics:

```
554 estpost tabstat potential_revenue assessment tax , by(year) stat(max)col(stat)
555 esttab . using "$xlsout/results_nominal.csv", cells("max(lab(Max) fmt(%15.2fc))")
↳ noobs nonumber eqlabels(`e(labels)') varwidth(20) replace
556
557
558 restore
```

Stage 4: Estimating the Tax Gap

4. Obtain Nominal Values for Potential Revenue, Total Tax Collected, and Potential Revenue Lost

B. Results by Sector/Industry/Region

Collapse Data and Calculate Potential Revenue:

```
562 preserve
563
564 collapse (sum) assessment tax if reg_audit==1, by(var_analysis)
565
566 generate potential_revenue = assessment+abs(tax)
567
```

Plot Revenue Consequences:

```
569 graph bar (mean) potential_revenue assessment tax, over(var_analysis) blabel(bar,
↳ size(vsmall) format(%9.2f)) legend( label(1 "Potential Revenue") label(2 "Revenue
↳ Lost (predicted)") label(3 "Revenue Collected") ) ytitle(Monetary Unit) title(
↳ "Revenue Consequences") subtitle("by audited tax regions") note("Aggregate values.")
570 graph save Graph "$graphs/revenue_gap.gph", replace
571 graph export "$graphs/revenue_gap.pdf", as(pdf) replace
572
```

Export Summary Statistics:

```
573 estpost tabstat potential_revenue assessment tax , by(var_analysis) stat(max)col(stat)
574 esttab . using "$xlsout/results_nominal.csv", cells("max(lab(Max) fmt(%15.2fc))")
↳ noobs nonumber eqlabels(`e(labels)') varwidth(20) append
575
576
577 restore
578
```

Concise Summary of the Bottom-Up Tax Gap Analysis Toolkit

Aim: Equip Revenue Authorities with a machine learning framework to estimate tax gaps for VAT, PIT, and CIT, enhancing tax compliance and policy formulation.

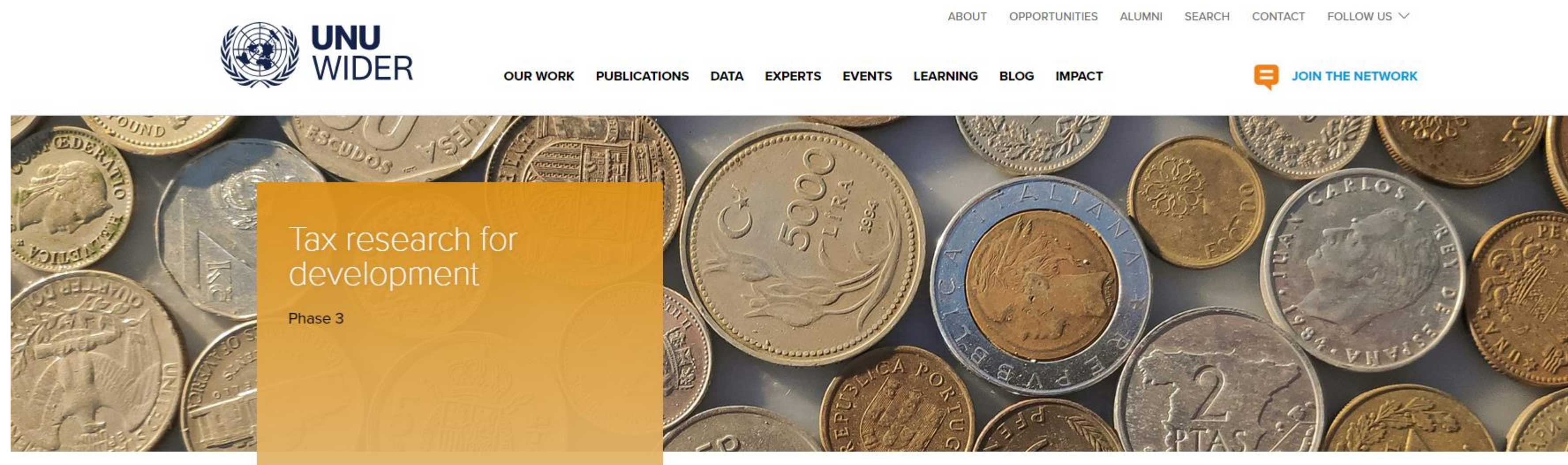
Process Overview:

1. **Setup:** Install required Stata packages and prepare the analysis environment.
2. **Model Selection:** Employ Random Forest algorithms, tuning parameters to optimize predictions of tax evasion using relevant variables.
3. **Model Testing:** Assess the model's performance with metrics like R-squared and RMSE; compare results with traditional OLS regression to ensure accuracy.
4. **Tax Gap Estimation:** Apply the optimized model to the full dataset to predict unpaid taxes, calculate tax gap percentages across different groups (e.g., year, industry, region), and address any negative values in the evaded tax variable.
5. **Revenue Analysis:** Calculate nominal values for potential revenue, total tax collected, and predicted revenue lost. Generate visual summaries and detailed statistics to illustrate revenue implications.

Outcome: Provide a data-driven approach to identify significant tax evasion areas, facilitating better compliance strategies and informed policymaking.

Bottom-up approach materials

- This workshop trains to use the Bottom-up approach. Providing technical sessions with examples of estimating the **Tax gap** using **dummy data**
- Tax Gap Toolkit **Technical Note**, **Readme files** and example **Stata codes** will be available on the UNU-WIDER's webpage later this year



Thank you!

Dr Kwabena Adu-Ababio

kwabena@wider.unu.edu



**UNU
WIDER**

More information available online

wider.unu.edu

Follow us on LinkedIn, Facebook, and Twitter **@UNUWIDER**