EXPLORING SEMI-SUPERVISED LEARNING METHODS FOR TAX FRAUD DETECTION USING PHILIPPINE TAX DATA

R DOCUMENTATION

MARKY ERWIN S. CHUA
PATRICK A. MOLL

The following are the list of functions in the file `ModelTrainingandValidation.R`

1. `filter_regdata_by_industry`
2. `filter_slsp_by_tins`
3. `get_sales_cols*`
4. `get_purch_cols*`
5. `aggregate_industry_sales`
6. `aggregate_industry_purchases`
7. `keep_latest`
8. `get_vat_cols*`
9. `get_ratio_sales_to_purchases`
10. `get_ratio_vat_to_sales`
11. `get_ratio_vat_to_purch`
12. `year_qtr_df*`
13. `add_year_qtr`
14. `cross_checking`
15. `add_with_values`
16. `update_with_latest`
17. `zscore_sales`
18. `zscore_purchases`
19. `benford_purchases`
20. `benford_sales`
21. `zscore_vat`
22. `create_indept_vars*`
23. `data_processing_for_training_set*`
24. `data_processing_for_validation_set*`
25. `data_splitting`
26. `umodel*`
27. `ssmodel*`
28. `validation`
29. `full_validation*`

`*` - Codes with this mark are considered the main functions, which are documented in this file.

year_qtr_df

Description

`year_qtr_df` creates a time table which contains a sequence of quarterly periods spanning from the specified start year and quarter to the specified end year and quarter, inclusive.

Usage

```
year_qtr_df (start_year, end_year, start_qtr, end_qtr)
```

Arguments

`start_year`	The first year to start counting from.
`end_year`	The final year where the counting ends.
`start_qtr`	The quarter of the `start_year` to start counting from
`end_qtr`	The last quarter of the `end_year` to still be counted.

Values

`year_qtr_df` returns a dataframe, wherein each row in the data frame represents a single quarter and includes two columns: `'tax_year'`, which indicates the year of the quarter, and `'qtr'`, which specifies the quarter within that year (e.g., 'Qtr 1', 'Qtr 2', etc.).

`get_sales_cols`

Description

`get_sales_cols` extracts specific columns from the SLS dataset.

Usage

```
get_sales_cols(old_sls)
```

Arguments

`old_sls`	The original sls dataset from which columns are to be extracted.

Values

`get_sales_cols` returns a new dataframe containing the following columns:

- `'owner_tin'`: The taxpayer identification number of the owner.
- `'tax_year'`: The tax year corresponding to the sales data.
- `'qtr'`: The quarter of the tax year for which the sales data is recorded.
- `'sls_taxable_sales'`: The amount of taxable sales recorded for the specified quarter.

get_purch_cols

Description

`get_purch_cols` extracts specific columns from the SLP dataset.

Usage

`get_purch_cols(old_slp)`

Arguments

`old_slp`	The original SLP dataset from which columns are to be extracted.

Values

`get_purch_cols` returns a new dataframe containing the following columns:

- `'owner_tin'`: The taxpayer identification number of the owner.
- `'tax_year'`: The tax year corresponding to the purchases data.
- `'qtr'`: The quarter of the tax year for which the purchases data is recorded.
- `'gross_taxable_purchases'`: The amount of gross taxable purchases recorded for the specified quarter.

`get_vat_cols`

Description

`get_vat_cols` extracts specific columns from a VAT dataset.

Usage

```
get_vat_cols(old_vat)
```

Arguments

`old_vat`	The original VAT dataset from which columns are to be extracted.

Values

`get_vat_cols` returns a new dataframe containing the following columns:

- 'DATE_FILED': The date when the VAT was filed.
- 'owner_tin': The taxpayer identification number of the owner.
- 'tax_year': The tax year corresponding to the VAT data.
- 'qtr': The quarter of the tax year for which the VAT data is recorded.
- 'NET_PAYABLE': The net amount payable for VAT.
- 'AMENDED_YN': Indicator of whether the VAT filing was amended (e.g., "Y" for yes, "N" for no).

`data_processing_for_training_set`

Description

`data_processing_for_training_set` prepares the training datasets for fraud detection modeling. It performs the data preprocessing steps for the training datasets.

Usage

```
data_processing_for_training_set(reg.f, sls.f, slp.f, ebir.f,
efps.f, reg.2, sls.2, slp.2, ebir.2, efps.2)
```

Arguments

`reg.f`	The registration dataset for the fraudulent training dataset.
`sls.f`	The SLS dataset for the fraudulent training dataset.
`slp.f`	The SLP dataset for the fraudulent training dataset.
`ebir.f`	The eBIR dataset for the fraudulent training dataset.
`efps.f`	The eFPS dataset for the fraudulent training dataset.
`reg.2`	The registration dataset for the unlabeled training dataset.
`sls.2`	The SLS dataset for the unlabeled training dataset.
`slp.2`	The SLP dataset for the unlabeled training dataset.
`ebir.2`	The wBIR dataset for the unlabeled training dataset.
`efps.2`	The wFPS dataset for the unlabeled training dataset.

Values

`data_processing_for_training_set` returns a processed training dataset for fraud detection modeling.

Details

This function aggregates industry-level sales and purchases data separately for the fraudulent training dataset and unlabeled training datasets. It combines eBIR and eFPS datasets for each period and converts quarter values to standard format (e.g., "Qtr 1", "Qtr 2"). It then performs data imputation and validation checks, including removing duplicates and checking for data consistency. After preparing the datasets, it assigns labels to each observation based on fraud status and updates the dataset with the latest information for each taxpayer. The final dataset is ready for use in training fraud detection models.

`data_processing_for_validation_set`

Description

data_processing_for_validation_set prepares a validation dataset for fraud detection modeling.

Usage

data_processing_for_validation_set(validation_set_label = FALSE, reg.v = FALSE, sls.v = FALSE, slp.v = FALSE, ebir.v = FALSE, efps.v = FALSE)

Arguments

`validation_set_label`	The label to assign to the data points.
`reg.v`	The registration dataset for the validation dataset.
`sls.v`	The SLS dataset for the validation dataset.
`slp.v`	The SLP dataset for the validation dataset.
`ebir.v`	The eBIR dataset for the validation dataset.
`efps.v`	The eFPS dataset for the validation dataset.

Values

data_processing_for_validation_set returns a processed validation dataset for fraud detection modeling.

Details

This function aggregates industry-level sales and purchases data for the validation dataset. It combines eBIR and eFPS datasets, converts quarter values to standard format (e.g., "Qtr 1", "Qtr 2"), and performs data cleansing and validation checks, including removing duplicates and checking for data consistency. If labels are specified, it assigns them to each observation in the dataset. Finally, it updates the dataset with the latest information for each taxpayer. The processed dataset is suitable for use in validating fraud detection models.

`create_indept_vars`

Description

`create_indept_vars` generates independent variables for analysis based after data processing has been performed on a data set.

Usage

```
create_indept_vars(valid.full, sls.full, slp.full, valid.ind.med)
```

Arguments

`valid.full`	The complete dataset that has already undergone data processing.
`sls.full`	The full SLS dataset.
`slp.full`	The full SLP dataset.
`valid.ind.med`	The dataset used in computing median values.

Values

`create_indept_vars` returns a dataframe containing the generated independent variables values of each company.

Details

Use this function after using the data processing functions. This function calculates independent variables for analysis using the provided datasets and median values for individual taxpayers. It computes z-scores for sales, purchases, and VAT data based on median and median absolute deviation (MAD) values, which is based on the `valid.ind.med` dataset. The resulting z-scores are then joined into a single dataframe. Additionally, it computes Benford's Law conformity measures for both sales and purchases datasets and appends them to the main dataframe. The final dataframe contains independent variables ready for further analysis.

`umodel`

Description

`umodel` trains the unsupervised learning on the “Known Fraud” dataset and uses the trained model to predict pseudo-labels for an unlabeled dataset.

Usage

```
umodel (train_fraud, unlabeled, u_method)
```

Arguments

`train_fraud`	The “Known Fraud” dataset used in training
`unlabeled`	The unlabeled dataset used in training
`u_method`	The unsupervised learning method used. Possible values are “iso” for isolation forest and “svm” for one-class SVM.

Values

`umodel` returns a dataframe containing the “Known Fraud” data and the pseudo-labeled data.

Details

This function trains either isolation forest or one-class SVM on `train_fraud`, a set of “Known Fraud” data, and then, uses the trained model to predict the labels of the dataset `unlabeled`. If `u_method` is set to `"iso"`, the function trains an isolation forest model with 200 trees and dimension 1 on `train_fraud`. If `u_method` is set to `"svm"`, the function trains a one-class SVM model. After training the chosen model, it then predicts the labels of an unlabeled dataset. For SVM, the TRUE label indicates that the company is “Likely Fraud” while the FALSE label indicates “Likely Legitimate.” For isolation forest, a score greater than 0.50 indicates that the company is “Likely Fraud”; otherwise, the company is “Likely Legitimate.”

`ssmodel`

Description

`ssmodel` trains the supervised learning on the pseudo-labeled training dataset from `umodel` and results in a trained supervised learning model which can be used to predict the labels of an unlabeled dataset

Usage

```
ssmodel (train_fraud, unlabeled, u_method)
```

Arguments

`u.data`	The pseudo-labeled training dataset
`s_method`	The supervised learning method used. Possible values are "multi" for multinomial logistic regression and "rf" for random forest

Values

`ssmodel` returns a trained supervised learning model.

Details

This function trains either multinomial logistic regression or random forest on a set of pseudo-labeled training data. The result would be a supervised learning model which can be used to predict the labels of an out-of-sample dataset.

full_validation

Description

`full_validation` performs the model building, and then performs model validation using different validation strategies.

Usage

```
full_validation(validation_type, main_full, main_v = FALSE, seeds = FALSE)
```

Arguments

`validation_type`	A character string indicating the type of validation to be performed. Possible values are "cross" for cross-validation and "fixed" for fixed validation.
`main_full`	The main training dataset containing observations for fraud detection modeling.
`main_v`	The validation dataset used for fixed validation. Default is FALSE, not needed for cross validation.
`seeds`	A vector of random seeds used for cross-validation. Default is FALSE, not needed for fixed validation.

Values

`full_validation` returns a list containing the results of the validation process for each model and validation type. Specifically, it returns a list containing the following values:

1. `` `df.svm.multi` ``: A dataframe containing the results of support vector machine (SVM + MULTI) model validation using multiple validation strategies. It includes the percentage of fraud (% of F), known fraud (% of KF), and labeled legitimate (% of LL) observations.
2. `` `df.iso.multi` ``: A dataframe containing the results of isolation forest (ISO + MULTI) model validation using multiple validation strategies. Similar to `df.svm.multi`. It includes the percentage of fraud (% of F), known fraud (% of KF), and labeled legitimate (% of LL) observations.
3. `` `df.svm.rf` ``: A dataframe containing the results of (SVM + RF) model validation using a random forest (RF) strategy.It includes the percentage of fraud (% of F), known fraud (% of KF), and labeled legitimate (% of LL) observations.
4. `` `df.iso.rf` ``: A dataframe containing the results of (ISO + RF) model validation using a random forest (RF) strategy. It includes the percentage of fraud (% of F), known fraud (% of KF), and labeled legitimate (% of LL) observations.

Each dataframe represents the validation results for the corresponding model and validation strategy.

Details

This function builds the models, and performs validation on them using different validation methods. If validation_type is set to "cross", it performs cross-validation using the provided seeds. If validation_type is set to "fixed", it performs fixed validation using the `main_v` dataset. The function returns the results of the validation process for each model and validation type.