Universiteit Leiden
The Netherlands

Opleiding Informatica

Predicting tax codes using machine learning

Marcus Abukari

Supervisor:

Dr. S. Verberne & Thomas Offerman

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
`www.liacs.leidenuniv.nl` 21-08-2018

Abstract

The Financial Shared Service Centre (FSSC) is responsible for all financial-administrative services within Leiden University. The FSSC already made advances using data analytics to support primary processes and reducing errors. This research project focuses on the classification of outgoing transactions with the correct tax code. The FSSC supplied a one year dataset of all outgoing transactions labeled with the correct tax code. Due to limitations of the used software stack, the dataset needed to be divided in a dataset containing only natural language, and a dataset containing categorical and numeric data. The experiments consisted of evaluating multiple machine learning algorithms on the supervised training data sets and combining the two best in to one model. The results showed that the Linear Support Vector Classifier performed best on natural language, and the Random Forest Classifier on categorical data.

Contents

Chapter 1

Introduction

1.1 FSSC

The Financial Shared Service Centre (FSSC) is the financial department within Leiden University. This department is mostly (but not exclusively) responsible for the processing and registering of university related invoices, staff declarations and salaries. This project is a collaboration between the FSSC and the Leiden Institute of Advanced Computer Science (LIACS).

1.2 Problem statement

Never before has it been easier to collect data than it is today. Technological advances over the past decades have increased storage possibilities while the relative costs for organizations dropped. Furthermore, the digitization and automatizing of business processes results in an increasing amount of data relevant to the organizational performance. This data can be used to improve performance at process level, and to gain a competitive advantage [3]. Even though generating and storing data proves relatively easy, using the data to ultimately obtain a competitive advantage is an aspect where origination often struggle.

In previous years, the FSSC made the first advances in using data analytics to prevent primary processes from being hindered by increasingly complex legislation which require business resources to implement and maintain. The goal is to reduce manual data input and validation, and replace this with automated data analytics. One of the more dynamic areas in this field is the classification of the value-added tax (VAT). Every transaction processed by the FSSC needs to be booked with the correct VAT code in order for the university to operate within the law, and for correct tax return application. The VAT codes can indicate charged or uncharged transactions, high or low rates and differ between national and international transactions.

In the current situation, classification is done by relatively simple conditional logic, and manually for the transactions that cannot be classified this way. One of the problems with this approach is that this only handles predefined rules, and already recognized problems. Therefore it misses rules that are not defined within the business and have to be solved manually.

1.3 Goal of this research

The main goal of this research is to build a classification model which is able to predict the tax code of a transaction. Besides the main goal, a secondary goal (formally outside of the scope of this project) is to implement this model in the existing data analytics system. Here, the model will be used to to solve the 2 business problems defined in the problem statement in the previous chapter:

- Automatically recognizing classification errors
- Reducing manual input and validation

For the model, the input is a transaction with all attributes of the transaction used for training the model, plus other master data that where merged in the transaction data. The output is one specific tax code. In practice it might be preferable to get more information about the prediction, since business decisions depend on this. In this case the model should calculate the probability of each prediction. If the predicted tax code of the model differs from the assigned tax code, and the probability is above a certain threshold, a flag should be raised.

To reach the main goal the following research question was formulated:

"Too what degree can machine learning algorithms predict the tax codes of FSSC transactions"

1.4 Thesis Overview

In the next chapter, prior research, general background information and definitions will be discussed. Chapter 3 will lay out the used data for this research project, as well as the preprocessing performed to optimize it for the classification algorithms. In this chapter, an abstract view of the used method is also discussed. In Chapter 4 the methods are put into practice and their corresponding results are listed. The research project ends with the conclusion in Chapter 5. Here will be defined if the results are acceptable from a academic point of view, as well as the relevance for the FSSC business practices.

Chapter 2

Background and related work

2.1 Tax codes

Value-added tax (VAT) is a type of indirect tax on the sale of products and services. The tax is levied on the sales price of products in each step of the production or distribution process. VAT is used in most developed countries with varying laws and rules. Even though the English term is used, this research project will always refer to the Dutch (or EU) VAT system unless otherwise specified.

Within the current Dutch VAT system there are three different tax rates [1]:

- 0%-rate: this rate is mostly used when exporting good from the Netherlands to other countries, as well as deliveries to ships and aircraft, fishing and excise goods.
- 6%-rate: this rate is also referred to as the low rate and is levied on common goods such as nutritional goods, medicine, books and agricultural products and services.
- 21%-rate: this rate is also referred to as the high or general rate. In October 2012 this rate was increased from 19% to 21%. Essentially, if products or goods are not except from VAT, do not fall in the 0% or 6% rate or do not fall under any other deviating rule, 21% VAT is calculated.

Besides the rates, there are also transactions exempt from VAT. Among others, goods that are exempt from VAT include healthcare services, education, financial services and insurances, sports organizations and fundraising activities.

The FSSC tax codes are a combination of three attributes. While the tax rates and VAT exemptions are defined by the tax authorities and used nationally, the FSSC tax codes are developed and used solely within the University. Besides the tax rates and the VAT exemptions the country is the third and final attribute of the FSSC tax code. The country refers to the country of the creditor and it distinguishes three possible situations:

- Creditors within The Netherlands

Table 2.1: Tax codes overview

Country	Charge	Rate	Tax code
Domestic	Uncharged	Low	P1
Domestic	Uncharged	High	P7
Domestic	Charged	Low	V1
Domestic	Charged	High	V4
Domestic	Charged	High	V7
Domestic	Uncharged	Low	X1
Domestic	Uncharged	High	X4
Domestic	Uncharged	High	X7
Domestic	Uncharged	High	P8
Domestic	Charged	High	V8
Outside EU	Charged	High	Y3
Outside EU	Charged	Low	Y5
Outside EU	Charged	High	Y6
Outside EU	Uncharged	High	Z3
Outside EU	Uncharged	Low	Z5
Outside EU	Uncharged	High	Z6
EU country	Charged	Low	Y1
EU country	Charged	High	Y4
EU country	Charged	High	Y7
EU country	Uncharged	Low	Z1
EU country	Uncharged	High	Z4
EU country	Uncharged	High	Z7
na	Uncharged	na	V9

- Creditors within the European Union
- Creditors outside the European Union

Table 2.1 provides an overview of all the combinations of the attributes with their corresponding tax code. Note that some tax code attribute combinations appear twice in the overview with different tax codes (X4 and X7 for example). This is due changes in dutch tax legislation that are in effect since 1 October 2012 [2]. This change increased the high tax rate from 19% to 21%.

2.2 Metrics

The goal of this research project is to develop a model which performs best at predicting the tax codes of FSSC transactions. There are several techniques to determine the quality of classification models, most of them depending on the use case of the model. These metrics mostly work with formulas containing true positives, false positives, true negatives and false negatives:

- A true positive (TP) is an outcome where the model correctly predicts the positive class. A true negative (TN) is an outcome where the model correctly predicts the negative class.
- A false positive (FP) is an outcome where the model incorrectly predicts the positive class and the false negative (FN) is, subsequently, the outcome where the model incorrectly predicts the negative class.

These examples assume that the target class is binary, either true or false. The dataset of this research project is a multiclass dataset with the ± 20 tax codes as classes. Here, the TP indicates that the tax code is correctly predicted while the FP refers to an incorrect prediction. The other outcomes (TN and FN) can be calculated by calculating the TN and FN separately for every class (tax code) and then averaging the results.

The most intuitive and straightforward metric is the accuracy score. The accuracy score reflects the subset or instances that are correctly predicted by the model and is formulated as follows:

$$Accuracy\ score = \frac{TP+TN}{N}$$

The accuracy score is a solid metric when the dataset and target classes are symmetric where they occurrence of each tax code is more or less equal for all. If the dataset however consists 90% of instances labelled with class A, and the algorithm predicts class A for every instance the accuracy score is 90% even though the quality of the model is not particularly high. As shown later in the exploratory data analysis, the dataset used is not balanced and the accuracy metric might not be the best metric to use.

An alternative to the accuracy score is the precision score. The precision metric is formulates as follows:

$$Precision = \frac{TP}{TP+FP}$$

The precision is the subset of positively labeled instances that were actually positive.

Finally there is the recall metric which is formulated as follows:

$$Recall = \frac{TP}{TP+FN}$$

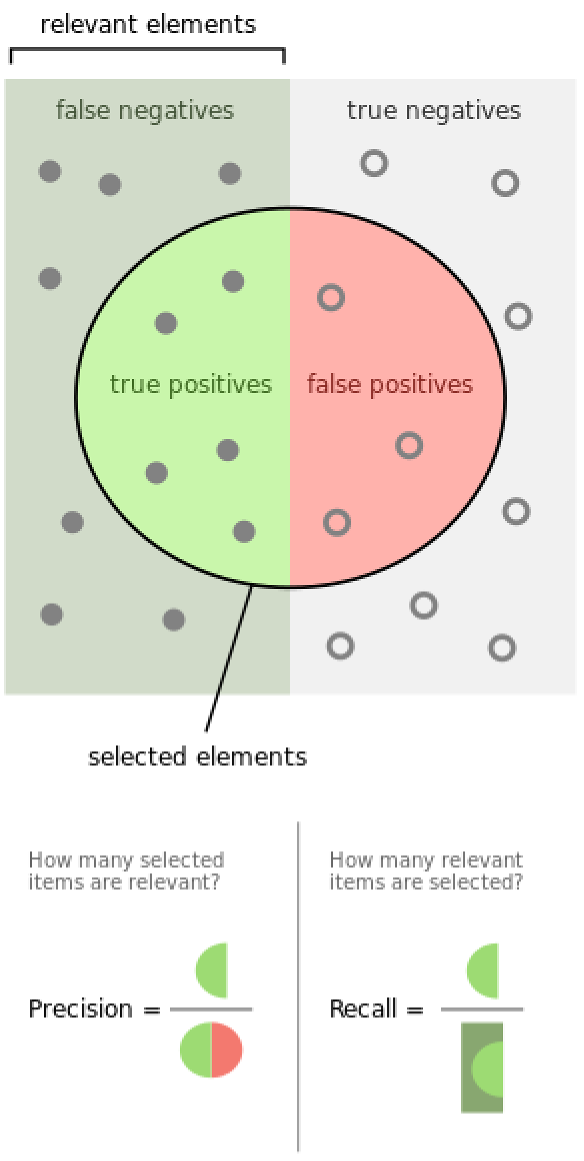
The recall is the subset of actual positive instances that were labeled as positive by the model. The difference between the precision and recall metric is visualized in Figure 2.1.

This project focuses mainly on maximizing the accuracy score but will also list the precision and recall metrics for each class separately.

2.3 Prior research

Machine learning, data mining and big data are all subject to a large amount research within the academic field since these are relatively new concepts with promising possibilities. Unfortunately, these is little public research available similar to the problems this research project attempts to solve. One study that is interesting is the “Automatic Classification of Bank Transactions” master thesis from the Norwegian University of Science and Technology [4]. This study focuses on classifying bank transactions and assign them with a predefined category such as “Automobile and Transport”, “Groceries” or “Hobby and Knowledge”. The data consists of some categorical and numeric data, but the main focus (and source of information) is the description. In the study, the text is interpreted beforehand with third party resources before training the model. This interpretation is done with the Brønnøysund Registry and the Google Places API, both being databases that

Figure 2.1: Precision and recall visualized



contain information about companies with the former strictly containing Norwegian companies. The research concluded that using third party resources for text classification had a significant impact on the quality of the classification model.

2.4 Used hardware and software

The used dataset contains all FSSC transactions to companies, individuals and other organizations. For obvious reasons, this is sensitive information and should be treated as such. The FSSC provided a laptop for conducting research and experiments keeping the data on FSSC property. Experiment results and time reports therefore all come from the same machine with the following specifications:

- CPU: Intel(R) core(TM) i7-6500U CPU @ 2.50GHz
- Memory: 16GB RAM

As discussed in previous section, the FSSC already made progress using machine learning and data analytics. Besides the main information system, the data analytics team developed a Python based application to handle all analyses and machine learning activities. This application connects to the main SAP based information system and uses the traditional broker worker combination to periodically run analysis tasks on the database. The team decided to use the Python free software machine learning library scikit-learn (sklearn) for machine learning tasks. The module comes with multiple tools to aid the development of data mining models such as preprocessing, cross-validation, performance metrics and baseline algorithms. This speeds up the development and avoids reinventing the wheel while still using industry standard methods.

The goal of the project is to find the best data mining technique for the highest accuracy score possible. Implementation is not within the scope and should not hinder the main goal, but is highly desirable from the FSSC point of view. The Python and sklearn module combination aligns with the predefined goal, due to the high-level characteristics, and FSSC preferences since integration in the existing data analytics framework is relatively painless. It is therefore the optimal software choice for the project.

Chapter 3

Data & Methods

3.1 Data source

The model developed in this research project is limited to credit transactions. This decision was made because the tax codes of the debit side are differently constructed which essentially makes for two different kind of problems. For the project, the FSSC supplied a four year dataset containing approximately $1,200,000$ credit transactions. This ranges from employee lunch declarations to monthly subscription fees and faculty buildings maintenance costs. Table 3.1 provides an overview of the structure of this dataset as exported by the SAP information system.

Besides the main dataset there are several other datasets related to every transaction. In the main dataset this extra information is represented by a unique identifier referring to a row in a different dataset. Unfortunately, SAP does not provide the possibility to merge these datasets before exporting, meaning the extra data is exported as separate dataset files. The FSSC identified 3 master data datasets that may contain information about the tax code:

- Ledger master data
  Each row in this dataset is a specific ledger account. All transactions of the same type/category are connected to the same ledger account.
- Creditors master data
  The creditor of each transaction is the person of organization the money of the transaction is owed to. In this dataset, information about the creditor like address and account information is stored.
- Order master data
  The order can be regarded as a project within the university. An order has an assigned budget and every transaction related to an order references the order master data.

Table 3.1: Transaction table attributes overview

Column name	Data type	Description
Tax code	String	Initially assigned tax code
Document number	String	Unique transaction identifier
Document date	Timestamp	Free field
Booking date	Timestamp	Date the transaction occurred
Entry date	Timestamp	Date the transaction was booked
Amount	Float	The amount of the transaction
Currency	String	The currency of the transaction
Allocation	String	Free field (mostly used for extra date)
Order	Integer	Order ID
PrCtr	Integer	Profit center ID
Ledger	Integer	Ledger ID
Pos.	Float	SAP specific attribute
Position	Integer	Unique identifier
RkSrt	String	Account type (either 'S' or 'K')
Reference	String	Optional free field
Document header	String	Optional free field (often extra description)
Text	String	Transaction description
Segment	Integer	Segment ID (faculty)
Short text	String	Ledger description
Creditor name	String	Name of creditor (person or company)
Debtor name	String	Name of debtor (not relevant for credit dataset)
Trans.code	Integer	SAP specific code
Username	String	Username of person that entered the transaction in the system
BdNr	Integer	Business unit ID

3.2 Data cleaning

For this project a total of 4 separate datasets are used. Besides the main dataset there are the three master data datasets described in the previous section. All the datasets are exported by the same SAP information system and therefore share the same challenges when it comes to reading, cleaning and preprocessing of the data. The goal of these operations is to transform the 4 files in to one pandas DataFrame that is free of any aspects that may hinder the algorithms in training a model.

The supplied datasets are regular text files in CSV format. The pandas module provides the powerful function `readcsv()` that, with it's parameters, enables it to read the file and format it properly in one call. The parameters utilized to clean to data while reading the CSV file are as follows:

- `sep`: the delimiter to use. This option defaults to a comma but in this case the datasets use a tab as a delimiter.
- `usecols`: the function allows for using only the columns defined in this parameters list. This is particularly helpful since there are multiple sequences of tabs in the data which pandas will read as an empty, and unnamed, columns.
- `index_col`: defaults to `True` but is not required for this data since there is no index column.
- `skip_rows`: the number of rows to skip beforehand, this differs per datasets from 3 to 5.

- `error_bad_lines`: the data exported by the SAP information system is not optimized for the pandas module and may conflict at some aspects. Besides that, there are some free fields where users are able to input data to their liking. This option makes sure that the script does not terminate when encountering unsupported symbols but instead skip the entire row.
- `skipinitialspace`: skips spaces after the delimiter
- `thousands`: overrides the default thousand separator. In this case it does so from the default '.' to ','.

Since the `readcsv()` function handles almost all cleaning, the only requirement left is dealing with the missing values in the data. The pandas dataframe object has the appropriate method, `dropna(how='all')`, built in. This methods drops any row or column where *all* values are missing skipping rows of column with partly missing values.

Lastly, the 4 separate dataframes need to be merged into one. The main dataset functions as the base where the other dataframes are merged into. Again, the dataframe object comes with the built in method `join()` to accomplish this result. The method, which is called on the main dataset dataframe instance, takes the new dataset as parameter accompanied with the name of the column that should be matched between the two dataframes. After all dataframes are merged, the resulting dataframe contains all existing data plus its ledger, creditor and order information.

3.3 Preprocessing

3.3.1 Natural language

At this point, the dataframe consists `X` rows (this depends on the sample size used in development) and `20+` columns or attributes. Even though there are multiple attributes with the data type `string`, there are certain columns that represent natural language. The column `Text` in the main dataset is one example of a natural language `string` field. This column contains a short description entered by the person that booked the transaction.

The natural language transformation algorithms of sklearn only support single columns. This means that all natural language columns have to be combined into one new column: `Combined text`. After merging there are three operations performed to reduce the noise in the text:

1. Capital letters are converted to lowercase characters
2. Numbers are removed from the text
3. Special characters (braces, semicolons etc) are removed from the text
4. All words containing less than 4 words are removed

Table 3.2: Cleaning natural language text

Raw text		Cleaned text
van dobben kroket rundvlees stuk 100 gra		dobben kroket rundvlees stuk
BKO maatwerkcursus aanvraag 1		maatwerkcursus aanvraag
smit uienringen, zak 1 kg (EH: ZK)	$\longrightarrow$	smit uienringen
bo grachtenbrood gesneden 18mm (EH: stuk)		grachtenbrood gesneden
HB/50001626 Flyer Thorbecke (100x)		flyer thorbecke

Table 3.3: Natural language columns

Column name	Dataset	Description
Document header	Main	Free field
Text	Main	Transaction description
Short text	Main	Ledger description
Long text	Ledger	Long/Extra description
Name 1	Creditors	Creditor name
Resp.person	Order	Responsible person
Applicant	Order	Applicant name
Name budget holder	Order	Transaction approver
Username budget holder	Order	Username transaction approver

Table 3.2 shows some examples of this process. After cleaning, the merged columns are removed from the dataframe.

3.3.2 One hot encoding

In order for the algorithms to train on the data, all data needs to be converted to numerical values. The natural language column `Combined text` is an exception to this which is explained further in the next subsection. The majority of the attributes besides natural language are `string` attributes with nominal data. Nominal data, or categorical data, is data with no inherent order or sequence. Examples of categorical attributes in this dataset are country, segment and currency. In this step of the preprocessing, these categories need to be converted to numerical data. The most obvious approach is to assign a number to each category which is demonstrated in Table 3.4.

While seemingly intuitive, this will lead to unexpected training behaviour since algorithms will interpret the values as being ordered. Referring back to Table 3.4, the algorithms will interpret the currencies as ordered meaning the Pound is "higher" than the Yen, and the Dollar as being "lower" than the Yen. The solution to this problem is One Hot Encoding. One Hot Encoding a column means converting every (unique) value of the categorical attribute to its own new column. The value of this new attribute is one bit indicating if the instance has this category as its value. Table 3.5 demonstrates the process of One Hot Encoding the currency data of

Table 3.4: Transforming nominal data to numeric

Currency		Currency
Euro		0
Dollar	$\longrightarrow$	1
Yen		2
Pound		3

Table 3.5: One Hot Encoding nominal data

Currency		Euro	Dollar	Yen	Pound
Euro		1	0	0	0
Dollar	$\longrightarrow$	0	1	0	0
Yen		0	0	1	0
Pound		0	0	0	1

Table 3.4.

3.3.3 Bag of words

The bag of words model is used to convert natural language to a representation that the machine learning algorithms can work with. The model shares the same characteristics as One Hot Encoding. Instead of converting every unique value of the categorical data attribute, every word in the text is converted to a new column. The value of this column is not a binary value but a integer specifying the occurrence frequency of the word. Given are three text instances:

- x1: Mark owns a house
- x2: Mark owns a car
- x3: Kim owns a house and owns a car

Table 3.6 shows the resulted table after the Bag Of Words model has converted the text instances.

In this new table, the number of features is the number of distinct words in the dataset which would grow quickly. Given a dataset with 100.000 distinct words (features) and 10.000 samples, storing this table in a Numpy array would require $10.0000 * 100.000 * 4$ bytes $=$ 4GB in RAM [5]. This is to large to work with and is solved by the sklearn module by using sparse matrices that only store the non-zero parts of the dataframe. This makes it possible to work with the dataset of this research project. The downside however is that the sparse matrix can not be combined with other (numeric) data.

When using the Bag of Words model it's common practice to combine this with Tf-idf term weighting. This method assigns weights to words according to their frequency in the entire dataset. Words with a high frequency are given a low weight which gives the algorithms an indication that the word contains relatively less information than words with a lower frequency. In practice this is helpful to prevent high frequency words that contain little information, such as "the", "a"or "in" from hindering the algorithms performance. In this case however, this problem is already prevented with the text cleaning and given the diversity of the text it's undesirable to penalize high frequency words.

Table 3.6: Bag of words model

X	Mark	owns	a	house	car	Kim	and
x1	1	1	1	1	0	0	0
x2	1	1	1	0	1	0	0
x3	0	2	2	1	1	1	1

Table 3.7: One Hot Encoding country with extra step

Country	Country code
The Netherlands	NL
Germany	DE
Japan	JP

$\longrightarrow$

Country code
NL
EU
OEU

$\longrightarrow$

NL	EU	OEU
1	0	0
0	1	0
0	0	1

3.3.4 Other preprocessing

The creditors master data dataset contains information about the whereabouts of the creditor. Particularly interesting is the country which stores the country code of each creditor in ISO 3166-1 format. As the tax code overview of Table 2.1 shows, the country is one of the factors making up the tax code. However, the only information required is if its domestic, within or outside the EU. Using just the country code is therefore less accurate and very inefficient when One Hot Encoding. Instead of One Hot Coding it directly, the data is first categorized in domestic (`NL`), in the EU (`EU`) and outside of the EU (`OEU`). Table 3.7 demonstrates this extra step.

Lastly, the dataset also contains an `Amount` attribute. This attribute does not contain missing values and is already cleaned by the `readcsv()` function which parses the thousand symbols. However, almost every transaction is listed twice in the dataset, to deduct from the ledger and to add to the creditors. This may confuse algorithms as a transaction with an amount of Therefore to optimize this attribute each value is converted to its absolute value. This also prevents excluding algorithms that only work with non-negative numbers.

3.3.5 Text data and numeric data

After the prepossessing the dataset contains two different types of data:

1. Natural language: this is the `Combined text` column containing all natural language. The column is converted to a sparse matrix using the bag of words model.
2. Numeric data: the only numeric column in the dataset was the `Amount` columns. The One Hot Encoding however transformed all the other (categorical) data to numeric columns. For consistency, in the rest of this report the `Amount` column combined with all One Hot Encoded categorical data will be referred to as numeric data.

At this point, almost all cleaning and preprocessing of the data is done. However, one final problem remains: the sklearn module and classification algorithms do not support the training of both text and numeric data at the same time due to the use of the sparse matrix. To solve this problem the data will be split in to different datasets (dataframes):

Table 3.8: Tax code frequency

Tax code	Frequency
V1	33.526%
V9	23.919%
X7	17.017%
P7	9.185%
V7	7.592%
P1	2.751%
Z7	1.933%
Z3	0.589%
X1	0.552%
Y7	0.413%
A9	0.280%
Y3	0.125%
Z1	0.108%
Z5	0.076%
A1	0.064%
A5	0.060%
A7	0.011%
P8	0.009%
V8	0.005%
Y5	0.004%
Y1	0.004%

- Text data: the sparse matrix containing all natural language.
- Numeric data: containing the `Amount` column with all categorical One Hot Encoded data.

3.4 Distribution of the categories

With the data read, cleaned and preprocessed, we conduct exploratory data analysis to get a better understanding of the data. The objective here is to identify attributes that have a high potential of aiding in the prediction of the target variable tax code, but also identify attributes that do not.

Table 3.8 shows the distribution of the target variable tax code. It becomes evident that this is not an even distribution. 7 of the 21 tax codes make up ±96% of the total dataset. This might prove it challenging for algorithms to learn the other tax codes since these instances will be sparse. This data also provides a baseline for the evaluation of the classification algorithms. This baseline is fixed at 34% since this would be the accuracy of a model that would always predict the most common tax code.

3.5 Approach

3.5.1 Combining models

The preprocessing resulted in two different datasets (dataframes). The two dataframes will both be trained with their own classification algorithm resulting in two different models that will probably predict different tax codes for some instances. In this project, we will look at three model combination methods for evaluating one final tax code:

- Single tax code guess: the first method consists of developing a model on the natural language dataset first. This model can then be used to predict the tax code for every transaction. The predicted tax code is added to the numeric dataset as a new column `Tax code guess`. Finally, the new column is treated as a regular attribute on which the second model is trained.
- Multiple tax code guess: the second method consists of training both datasets types separately, returning two models and two tax code guesses for each transaction. For each transaction the two predictions are added to the (original) numeric dataset. For this dataset, a third and final model is developed which produces the final prediction.
- Probability method: the third and final method is to make use of the `probability` parameter of the classification algorithms. With this option enabled, the models will, beside the tax code guess, also return the probability, or confidence, of the guess. The tax code prediction with the highest probability between the two models will be the transactions final prediction.

3.5.2 Comparing algorithms and parameters

The sklearn module comes with numerous classification algorithms built in. This introduces the challenge of finding the best algorithm for the two datasets. Some algorithms have specific strengths and use cases which makes them either more or less attractive to use. However, since there are a manageable amount of algorithms in the sklearn module, all of the algorithms that accept the data will be evaluated. Each algorithms will be evaluated using cross validation and compared on their corresponding accuracy score.

This will yield two best performing algorithms: one for the text data and one for the numeric data. The next step is to find the best parameters for the algorithms, in other words: parameter optimization. Here the sklearn provides the `GridSearchCV` function. This function takes a range of possible parameters and evaluates the accuracy score on the training set for each possible combinations of parameters. The function also cross validates this to prevent optimizing the parameters purely on one specific training set. Eventually, it returns the combination of parameters which results in the highest accuracy score.

The FSSC specified that they wish to gain insight in the rules the machine learning algorithms follow. One of the algorithms perfect for this requirement is the Decision Tree Classifier. When trained, the classification flow of the algorithm can be exported and analyzed. The main goal of this research project is to develop a model which can predict the tax codes of the FSSC transactions with the highest accuracy as possible. However, given the FSSC wishes, we will also develop a model using the Decision Tree Classifier provided that this method return acceptable results. The Decision Tree Classifier is not suitable for high-dimensional data as it would result in an unreadable tree, defeating the purpose in this case. The classifier will therefore only be used on the numeric data.

3.5.3 Target attribute split

As shown in Table 2.1, the tax code is essentially a combination of the three attributes country, charge and rate. The current approach describes the classification of this combinations result directly. However, this project will also compare the differences between predicting the tax code directly and predicting the tax code attributes beforehand. This means using the same two step approach described in Section 3.5.1 to predict every target attribute separately. The resulting 3 predictions will manually be converted to the corresponding tax code.

Chapter 4

Evaluation

4.1 Experiment structure

In this section multiple experiments will be conducted to develop classification models that can, as accurately as possible, predict the tax code of FSSC transactions. As mentioned in section 3.5.1, the two different datasets will be trained separately and will, most likely, develop models using different algorithms. Even though the resulting models differ, the two experiments follow roughly the same structure:

1. Take a (random) sample of the dataset
   The entire dataset consists of all FSSC transactions spanning a four year time period. This is a dataset containing over 1 million entries which is to large for experimenting. For both the text data and the numeric data the experiments are conducted on a random sample of $10,000$ transactions. This size makes it possible to train and test multiple algorithms and parameter options without losing to much information.

2. Split the sample in a training and test set.
   Training and testing on the same dataset is a methodological mistake. The trained model will predict the tax code just as seen during training resulting in a perfect test score. However, the model will struggle on new and unseen data rendering it useless in practice. The sklearn function `train_text_split()` splits the data randomly in a training and test set with a respective ratio of $80/20$.

3. Evaluating every possible/suitable model on the training data.
   The sklearn modules provides several built in and ready to use algorithms. In this part of the experiment, the goal is to find the best algorithm for the given dataset with best being defines as the algorithm that produces the model with the highest prediction accuracy score. Besides the regression algorithms, which obviously do not apply for this problem, there are several algorithms that have strict requirements that the data does not meet. This results in a subset of (depending on the dataset) approximately ± 10 algorithms. Intuitively, one might make an educated guess on the effectiveness of certain algorithms

Table 4.1: Text classification accuracy score results

Classification algorithm	Accuracy score	Evaluation time
Linear Support Vector Classifier	84.78%	2.054s
SGD Classifier	84.41%	1.279s
Random Forest Classifier	80.64%	10.775s
Decision Tree	79.89%	3.290s
K-nearest Neighbors	79.74%	5.374s
Multinomial Naive Bayes	77.27%	1.025s
Bernoulli Naive Bayes	72.31%	1.050s
Ada Boost Classifier	51.75%	6.713s

given the characteristics of the dataset and algorithms itself. For this research project however, it's decided to test every possible algorithm since the resources (most importantly training time) allow for this. To prevent overfitting, the accuracy scores of the algorithms are evaluating using cross-validation with 5 folds.

4. Parameter optimization for best algorithm(s)
   The algorithms used to develop models in the previous step do so with their default parameters. These parameters are predetermined by the sklearn development team and are defined as "sensible" parameters for most datasets. In this final step, parameter optimization will be performed using the `GridSearchCV` function described in section 3.5.2. The parameter optimization will only be performed on the best algorithm returned from the previous step. If, however, the difference between the top 2 or 3 algorithms is negligible, parameter optimization will also be conducted on these algorithms.

This part of the experiment, when conducted on the two different datasets, returns the two models with optimized parameters that are best in predicting the tax code of their dataset. Later during the experimenting, these are also the models used for the comparing and merging (unless otherwise stated).

4.2 Step 1: text data classification

The first experiment consists of the classification of the natural language (text data). During the preprocessing of the data, explained in depth in section 3.3.1, all of the natural language columns are merged into one. This preprocessing results in a two column dataset with the columns `Combined text`, containing all natural language, and a column with the target variable `Tax code`. We begin with following the experiment structure defined in section 4.1. After step 1, splitting the dataset in a training and test set, step 2 is performed: evaluating all possible algorithms on the dataset.

Table 4.1 lists all the classification algorithms used on a random sample of this dataset with their corresponding accuracy score and time used in seconds (this is the total time used for the 5 fold cross-validation). With an accuracy score of 86.1% the Linear Support Vector Classifier (LinearSVC) performs best and is followed, with some distance, by the SGD Classifier.

As for the LinearSVC parameters, the options are relatively limited. The only useful parameter with the goal of

Figure 4.1: LinearSVC parameter optimization

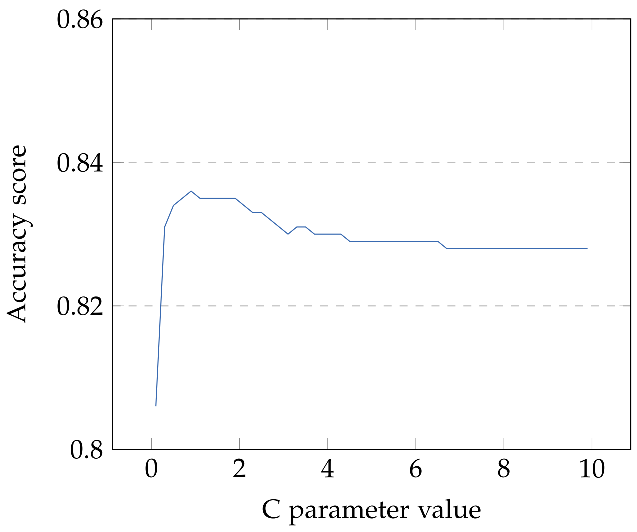


Table 4.2: Parameter optimization for the Linear Support Vector Classifier and the SGD Classifier

Classification algorithm	**Original accuracy score**	**Optimized accuracy score**
Linear Support Vector Classifier	82.85%	84.33%
SGD Classifier	82.80%	84.27%

improving the accuracy score is the C parameter. Figure 4.1 shows the effect of the LinearSVC's C parameter on the accuracy score. The graphs shows that a C of ± 0.5 results in the highest accuracy score.

During development and experimenting it became clear that in some cases the correlation between sample sizes and the algorithms accuracy scores would differ between algorithms. To conclude the experiment on the text data, the four best scoring algorithms are evaluated on different sample sizes. Figure 4.2 shows the result of this comparison. As seen, the increased accuracy score with bigger sample sizes are (for these algorithms) roughly equal to each other.

4.3 Step 2: numeric data classification

The numeric classification follows the exact same structure as the text classification in the previous section and thus starts with evaluating the performance of all possible algorithms. Table 4.3 lists all algorithms used on the dataset with the exception of those that perform under the accuracy score baseline of 34%. In general, we see the algorithms performing relatively poorly, especially compared to the results from the natural language mining. The Random Forest Classifier performs best with an accuracy score of 66.76%. Referring back to section 3.5.2, besides the Random Forest Classifier the Decision Tree Classifier, which has an accuracy score of 66.22%, will also be further developed in the parameter optimization.

Table 4.4 lists the parameter optimization for the two evaluated algorithms. While the Decision Tree Classifier performs slightly better, there is a considerably performance increase visible from the Random Forest Classifier.

Figure 4.2: Comparing the best algorithms on the text data with different sample sizes

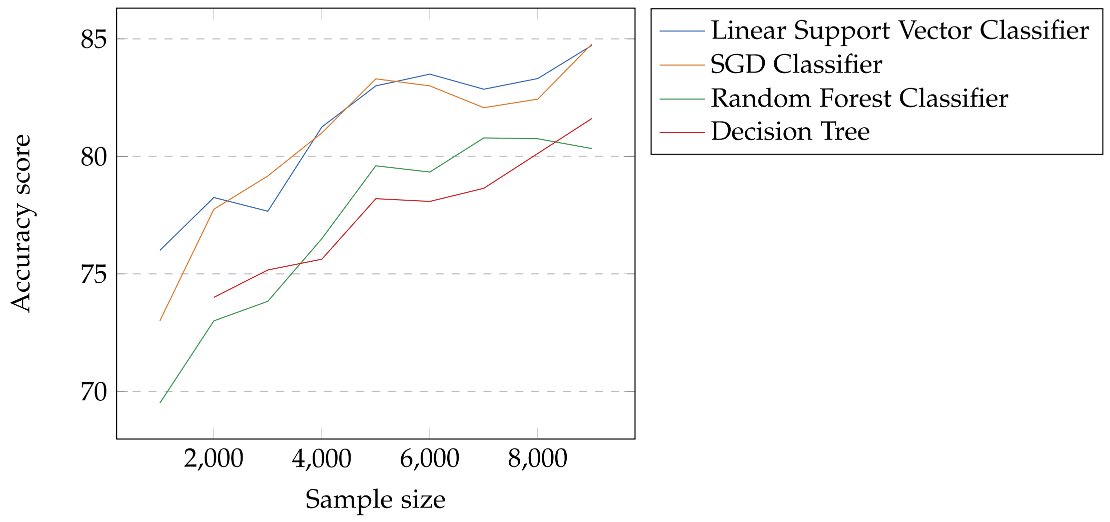


Table 4.3: Classification results on numeric data

Classification algorithm	Accuracy score	Evaluation time
Random Forest Classifier	66.76%	0.972s
Decision Tree	66.22%	0.340s
Bernoulli Naive Bayes	65.95%	0.169s
K-nearest Neighbors	58.69%	0.968s
Linear Support Vector	57.17%	32.188s
Ada Boost Classifier	51.24%	10.374s
Multinomial Naive Bayes	42.08%	0.092s
Gaussian Naive Baye	36.00%	0.501s

Table 4.4: Parameter optimization for the Random Forest Classifier and the Decision Tree Classifier

Classification algorithm	Original accuracy score	Optimized accuracy score
Random Forest Classifier	66.97%	71.01%
Decision Tree Classifier	67.23%	69.41%

Figure 4.3: Decision tree parameter optimization

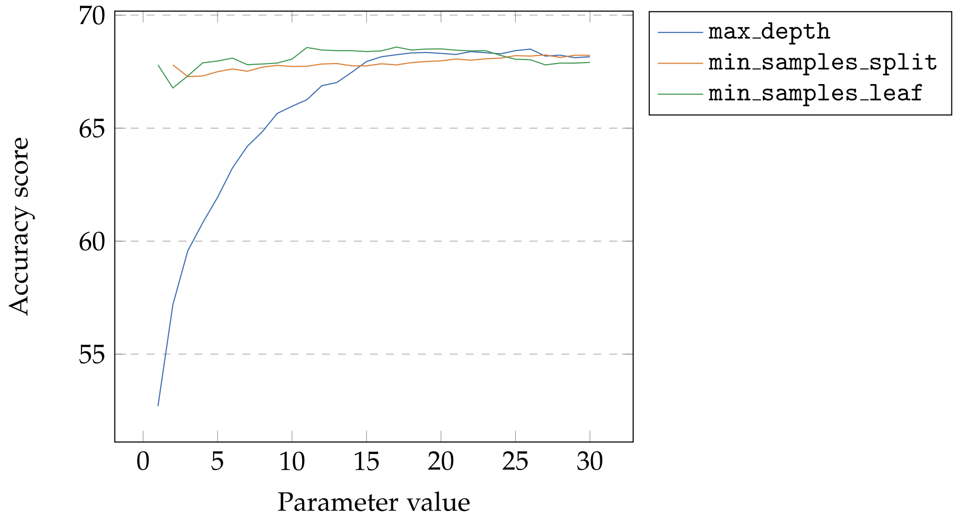


Table 4.5: Parameter optimization for the Random Forest Classifier and the Decision Tree Classifier

Experiment	Accuracy score
Categorical dataset model	72.70%
Natural language dataset model	84.03%
Single tax code guess method	93.20%
Multiple tax code guess method	92.26%
Prediction probability method	85.30%

4.4 Step 3: combining models

The previous experiments resulted in two models for both dataset types. These will be referred to as the numeric model and text model. The last part of the experiment consists of combining these models to utilize the combined prediction capabilities in order to increase the overall prediction accuracy. As described in 3.6 it's not possible to directly merge the two classification algorithms into one new model. However, three different, more manual, techniques are evaluated. The first option is to pass the tax code prediction of one model to the training data of the other. This begins with training and optimizing a model using the Linear Support Vector Classifier on the text data. The tax code prediction for each transaction is then added to the corresponding transaction in the numeric dataset. The numeric dataset is then subsequently trained and optimized, but now with the tax code as extra column. The resulting accuracy score of this model will the final result.

The second option evaluated here is predicting the tax codes separately and adding both of the predictions to the numeric dataset. A third model, will be trained using the two predictions as regular attributes to find the final prediction.

The last method is to use the prediction probability. For each transaction, the sklearn module provides the option to return the probability of each prediction. Here, both models will predict the tax codes for all transactions and the prediction with the highest probability will be the final prediction. The accuracy score will evaluated after every transaction has an assigned tax code.

Table 4.5 lists the accuracy score for both models separately, accompanied with the results of the model combining techniques. The accuracy scores differ slightly from those in the previous experiments due to a new (randomly selected) sample size. With a $\pm 1.27\%$ accuracy score increase, the prediction probability methods barely preforms better than the natural language model. The single tax code guess however increases the accuracy score by 10% to 94.14%and is the best option for combining the two models.

Table 4.6 shows the classification report for the single tax code guess method. This experiment was conducted using a sample size of 200,000 in order to show the results for all tax codes, and to more closely resemble real life performance.

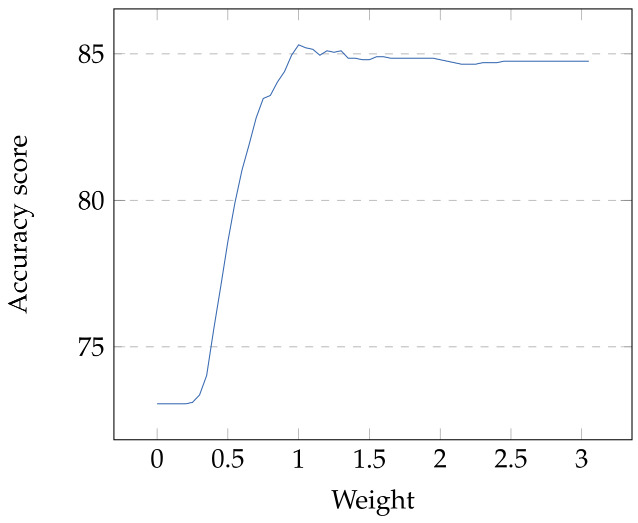
Since the prediction probability performs so poorly, it leads to wonder if there is any improvement possible in the function that determines the final tax code prediction of every transaction. Initially, the function just return the highest tax code probability. However, it is worth evaluating how weighted tax code probabilities affect the accuracy score. Figure 4.4 visualizes different weights used on the natural language models probability

Table 4.6: Classification report

Tax code	Precision	Recall	F1-score	Support
A1	0.91	0.50	0.65	20
A5	0.94	1.00	0.97	15
A7	1.00	1.00	1.00	5
A9	0.93	0.68	0.78	111
P1	0.86	0.85	0.85	1137
P7	0.89	0.87	0.88	3659
P8	0.00	0.00	0.00	3
V1	0.98	0.96	0.97	13451
V7	0.76	0.81	0.78	3028
V8	0.95	0.96	0.96	9567
V9	0.98	0.85	0.91	207
X1	0.94	0.98	0.96	6734
X7	0.00	0.00	0.00	2
Y1	0.94	0.62	0.75	50
Y3	0.00	0.00	0.00	3
Y5	0.79	0.68	0.73	164
Y7	0.85	0.81	0.83	43
Z1	0.90	0.82	0.86	243
Z3	0.74	0.96	0.84	24
Z5	0.88	0.89	0.89	771
Z7	0.91	0.91	0.91	53
avg / total	0.93	0.93	0.93	39290

prediction. The function `max(categorical_prediction, text_prediction)` is extended with this weight option becoming `max(categorical_prediction, text_prediction * weight)`. The figure starts (at a weight of 0) with the categorical models accuracy score, ends with the score of the text model, and shows a maximum at the default weight of 1 thus concluding that different weights do not increase the accuracy score.

Figure 4.4: Using weighted text classification predictions



Chapter 5

Conclusions

This project was commissioned by the FSSC with the goal of researching the possibility of automatic classification of FSSC credit transactions. For this research project, this resulted in answering the following research question: *"Too what degree can machine learning algorithms predict the tax codes of FSSC transactions"*. The result of this research is a method containing two machine learning models that when combined can predict the tax codes with an average accuracy score of 93%.

One of the more obvious use cases for this model is to detect errors after the initial classification by the FSSC's conditional logic. The data analysis team of the FSSC has already developed a framework that is able to connect to the SAP system and run data analyses periodically in which this model can be incorporated. In practice, this would result in an daily, weekly or monthly analysis of all recent transactions and would return transactions with conflicting tax codes. One limitation of the model in the current state is that the probability of the prediction is not returned. This could result in an unwanted amount of conflicts in cases where the model itself in unsure about the prediction. Implementing the probability prediction provides a certain threshold value and could be implemented as follows: "if the prediction differs from the assigned tax code and the probability of the prediction is higher than x% (where x is the desired threshold value), notify the business".

Other uses cases, which are more difficult to realize, include a combination of the machine learning model and the current classification method. In this method transactions that are not classified by the conditional logic, will be given a tax code by the model.

Bibliography

[1] Belastingdienst. Vat rates.

[2] Belastingdienst. Goods and services with 21% vat, 2012.

[3] Nadine Côrte-Real, Tiago Oliveira, and Pedro Ruivo. Assessing business value of big data analytics in european firms. *Journal of Business Research*, 70, 2017.

[4] Olav Eirik Ek Folkestad and Erlend Emil Nøtsund Vollset. Automatic classification of bank transactions. Master's thesis, Norwegian University of Science and Technology, 2017.

[5] Scikit-learn. Extracting features from text files, 2017.