Distinguishing self-built houses from project-built in new housing construction

A data science approach. Final version october 2019.

Management summary

In a feasibility study the Statistics Netherlands (*CBS*) and the Dutch national Cadastre (*Kadaster*) cooperated to investigate the possibility to detect self-built cases in new-built homes. The first reason to do this was to exclude self-built dwellings from the weight of the Price index for Newly built Dwellings (PND) in the European Owner Occupied Housing (OOH) as published by CBS. The second reason was to exclude self-built dwellings from the number of newly built dwellings, published also by CBS each quarter as a House Sales Indicator (HSI).

This report describes the project and explains the methods through which these self-built houses are automatically identified, including their feasibility. The main research questions of this study were:

- Are purchase deed texts a good source of information for training a predictive model?
- Which model types are best suited for taking advantage of the available data?

We trained a predictive model on the purchase deed texts available at the Kadaster. To do so, we needed two extra parts of information: an extensive list of addresses of newly built dwellings, in order to be able to create a training set with the corresponding deeds, and for each of these addresses annotated classifications, to be used in training the model. Both were gained by using the already collected data by CBS for measuring the PND. The final training set consisted of 14,180 dwellings built in the period 2015-2018. 12,832 (90,5%) were project-built and 1,348 (9,5%) self-built cases. Based on this, the Kadaster collected the corresponding deeds. Next we extracted its text. After that, a few models were trained, with a binary term vector model as the best performing.

The binary term vector model reached a 98,5% accuracy score in predicting self-built or project-built house type. This is a 8,5% percent point enhancement over the majority vote (90%). Given the fact that a relative simple binary term vector model reaches this score suggests that purchase deed texts are a good source of information for training a predictive model.

For future work on this topic several improvements on the model are proposed, as it has room left for improvement. Also implementing the proposed methodology into the current production process of the above mentioned statistics raises questions on the practical feasibility. Future work is required to answer these questions.

Introduction

Each quarter, Statistics Netherlands (CBS) publishes several house sales indicators, amongst which the number and value of all transacted newly built dwellings. The data needed is provided by the National Cadastre (Kadaster).
Late 2018 Statistics Netherlands (CBS) and the National Cadastre (Kadaster) took on the task of realizing an additional breakdown for transacted newly built dwellings in the Netherlands into two classes: project-built homes and self-built homes. The main use of this classification is twofold. First to exclude self-built dwellings from the weight of the Price index for Newly built Dwellings (PND)in the European Owner Occupied Housing (OOH). Second to exclude self-built dwellings from the number of newly built dwellings, published each quarter as a House Sales Indicator (HSI)[1] . The purpose of this

[1] https://www.cbs.nl/en-gb/our-services/methods/surveys/korte-onderzoeksbeschrijvingen/house-price-index--hpi---2015-100

report is to describe the project and to explain the methods through which these self-built houses are automatically identified, including their feasibility.

In order to be added to the OOH price index, the PND is constructed in accordance with the definition by Eurostat. This requires a price index based on project-built houses, i.e. housing projects that are planned, designed and constructed by a project development contractor and subsequently sold to individual private owners. These are different from self-built dwellings. In the simplest terms, self-built dwellings are “built by households”[2], and a self-builder is “someone that [*sic*] acts as his own developer".[3][4] The prototypical case of a project-built house is where an entire street or even neighbourhood is designed and built by a project development corporation that sells off the houses one at a time to individual private owners. In the prototypical case for a self-built house the private owner purchases solely an empty lot, consults an architect to design a house and subsequently contracts a construction company to build the house.

Currently CBS is not able to exclude self-built dwellings from the weight of the PND, nor from the HSI. This is because the self-built versus project-built attribute of a building is not part of any official parcel or building registration, this attribute was hitherto not known in the Netherlands. The goal of this project is therefore to explore the feasibility of an automated system that is able to detect all self-built cases in the pool of recently built homes. We do so by developing a predictive model that allows accurate estimation of the self- or project-built status of a house, based on the available knowledge of this house and its surroundings. As will be explained further on, we tried to use purchase deed texts for this as the primary data source. The main research questions are:

- Are purchase deed texts a good source of information for training a predictive model?
- Which model types are best suited for taking advantage of the available data?

Research design

The main aim of this study is to assess the best applicable data and the best performing model types for discerning project-built from self-built houses. We assume the best predictions can be gained from a model trained on the purchase deed texts since the deed more than occasionally states to what purpose the land parcel was bought. Therefore we trained a predictive model on these texts. To do so, we needed two extra parts of information:

- An extensive list of addresses of newly built dwellings, in order to be able to create a training set with the corresponding deeds.
- For each of these addresses annotated classifications, to be used in training the model.

Both were gained by using the already collected data by CBS for measuring the PND. Based on this, the Kadaster collected the corresponding deeds. Next we extracted its text. After that, the model was trained.

Data collection

Survey data collection

As stated, the sample of dwellings for this feasibility study was collected by CBS. Or rather we reused the existing survey for the PND used to collect quarterly price data on sold newly built dwellings. The process of

[2] Eurostat, *Technical manual on Owner-Occupied Housing and House Price Indices*, 2017, p. 100
[3] Idem, p. 30
[4] Idem, p. 22

survey data collection involved selecting project developers and asking them each quarter to fill in a list of house addresses that have recently been constructed by them. For each address the project developer fills in details regarding the contract, including the attribute in which category the built falls. They survey contains six categories defining project-built homes in one category and five degrees of self-built homes in the other categories. The survey defines self-built as "privately commissioned": "forms of housing development where the buyers purchase or lease the building lot themselves and choose an architect in order to maximize their influence on the realization of their home".[5] The survey is submitted to CBS where it is checked, cleaned and added to a central database.

The predictive models are trained using the accumulated results of this survey conducted in 2015, 2016, 2017 and 2018. The survey yielded 15930 responses with a valid address, of which 14384 respondents indicated their home was project-built (90,3%) and 1546 cases were self-built (9,7%). From these numbers, about 9 in 10 cases the records could be linked to actual addresses and land parcels. There are, however, a few caveats with regards to the final data set:

- The information given by the project developers is - in the absence of other sources - the best data source available, but it is not expected to be a gold standard. We can only assume that the respondents have read the instructions and definition for self-built or project-built housing included in the survey. It is almost guaranteed that there are some errors in the data labels, yet we do not know how many. It is likely, however, that a model accuracy of 100% is unattainable due to some unknown level of error in the survey responses.
- The level to which the survey data reflects the actual data distribution over the entirety of the Netherlands is a latent variable, again simply because the ratio of self-built versus project-built houses is not known historically.We take slight deviations of data set feature (or data property) distributions with regards to the national distribution as granted. The research that we present here is a feasibility study, not a production-ready method. Predictions made on data that was not included in the survey data set may be off and need to be re-checked for accuracy.
- Although the survey included a more differentiated classification system, the research is scoped to a binary classification of project-build versus self-build housing.

Spatial distribution

We are aware that there is some bias in the geospatial distribution of the classes in our data. While project-built homes are virtually ubiquitous, the self-built cases are clustered to some degree, due to policies in different municipalities. Some municipalities have designated certain neighbourhoods for the purpose of self-built homes. Using two maps we show the geospatial distribution of our classes in Figure 1 and Figure 2. The primary reason for choosing a heat map to represent the spatial distribution is because of local clustering of project-built homes. When supplying recently sold construction projects, construction companies often deliver a set of dwellings in project-built cases. Self-built

[5] https://www.cbs.nl/-/media/cbsvooruwbedrijf/prijsindex%20nieuwbouw%20koopwoningen/nieuwbouw_koopwoningen_4ekwartaal2018.xlsx?la=nl-nl

homes are, on the other hand, more often dispersed since most of these homes are often detached buildings. A simple instance plot therefore would show an over-representation of self-built homes, where these are actually under-represented in the data. The heat maps compensate for this disparity.

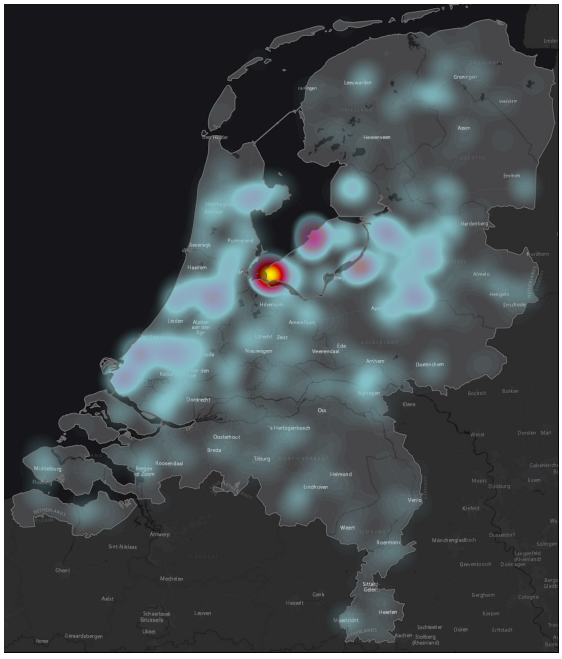

Figure 1: Heat map of self-built homes

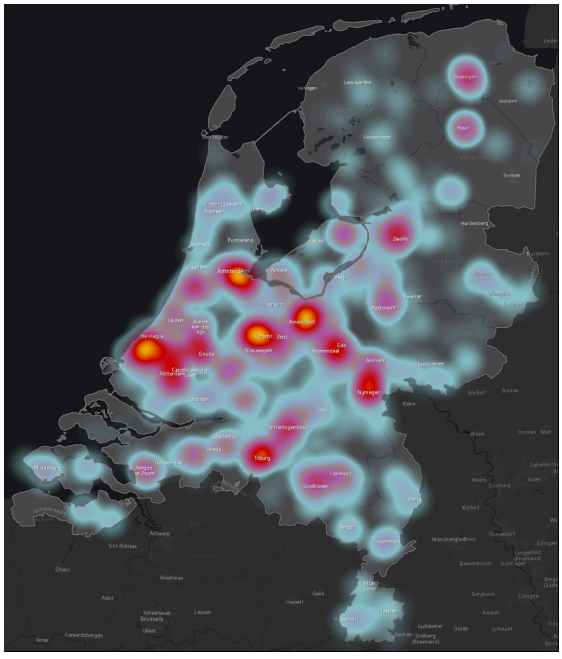

Figure 2: Heat map of project-built homes

Looking at the spatial distribution of self-built homes, it is obvious that there is a spike of activity in the center of the Netherlands in the city of *Almere*. Almere has a particular neighbourhood designated for self-built homes, the Oosterwold neighbourhood. When evaluating our model, we test the model for unwanted defaulting to a "Oosterwold-detector", choosing geospatial bias over other characteristics.

The original survey did include some basic address information such as street names, postal codes, house numbers and additions, place name and build type label. These were collected as feature collection A, and used to check the influence of geospatial bias on our model results.

Feature collection A: attribute data

Feature collection A can be regarded as a "classical" approach to predictive modelling: collecting and engineering attributes or *features* of the data instances that are possibly of value for modelling the specific task. These include a combination of single-label categorical data such as the house type and ordinal data such as the surface area of the parcel lot. In this particular case, we collected a Feature set A1 of the following attributes:

- **contract date (date)**: The date on which the contract with the construction company was signed.
- **postal code (categorical)**: The postal code for the building address. Since there are certain neighbourhoods in the Netherlands designated for self-built projects, a house within a certain postal code range has a higher likelihood of being self-built.
- **house number details (categorical)**: The house number, house letter (if

available) and house number addition (if available) for the building address. House letters often indicate an apartment, decreasing the likelihood of a self-built construction.

Feature collection T: text from deeds

Next purchase deed texts were linked to the survey data through two methods. Addresses are comprised of four different attributes: a date field (indicating the date at which the house was purchased) the street name, place name, postal code, house number, house letter and house number addition. When joined by hyphens the essential address components are known as a *PHT*: an abbreviation for the combination of postal code, house number, house letter and house number addition. These PHTs provide the primary means for lookups in the cadastral registration where this same PHT can be found. The database table containing these PHT entries also directly provide identifiers for cadastral registration objects and purchase deed texts. A second lookup method is through the Buildings and Addresses base registration identifiers, that provide lookup keys to the cadastral registration. A third method was supplied through the geolocation service[6] provided by the shared geoservice organisation for Dutch government (PDOK). This geolocator provides fuzzy search for addresses, allowing to add an extra 747 records from the survey (4.7%) to the final data set. After linking to retrievable purchase deed texts, a total of 14,180 of the 15,930 records (89.0%) from the survey was available, with a resulting total of 23,065 linked cadastral deed records. Of these linked deeds, 15239 are unique deeds, meaning that many deeds are linked to multiple addresses because they originate on the same parcel, or because the parcels and addresses appear in the same deed. Of the 14,180 usable records, 12,832 (90,5%) are project-built, with 1,348 (9,5%) self-built cases.

However not every linked cadastral record may be relevant to our task. The next phase was the issue of filtering out cadastral records that are dated too far off from the purchase date in the survey. The dates in the survey range from May 2015 through November 2018, where the dates in the recorded deeds from the cadastral database range from January 1992 through January 2019. The chance that a purchase deed from 1992 will yield useful information for a house built in 2015 is negligible, while the chance that too many irrelevant documents will interfere with model performance. Therefore, we needed to consider pruning our purchase deed texts to records that are likely to be relevant to the surveyed addresses. A reasonable range could be to allow the deed date to be within several months of the contract with the construction company. In order to assess a reasonable cutoff point, we created a distribution plot of the difference between the cadastral record date and the survey record date for every text we were able to link to the surveyed addresses. In the typical case, we expect a deal to have been struck between buyer and constructor before the details of this deal are committed to paper by a notary and sent to the cadastral registry. Thus, we chose to express date differences as positive when the survey (construction) date precedes the purchase deed date.

The histogram of the difference in days between survey date shows a negative-skew distribution with a (rounded to whole days) mean of -176 days but a median of -39 days. With a standard deviation of 781 days (more than two years!) the distribution has a considerable spread. The skewness can be explained by the fact that our retrieval is done

[6] Documentation in Dutch provided at https://github.com/PDOK/locatieserver/wiki/Zoekvoorbeelden-Locatieserver

on the basis of an address and subsequently a land parcel identification. From this query all deeds that match cadastral parcel identifier through the address are retrieved, which can be considerable for houses that are re-built from scratch on an existing parcel that had many previous owners, or apartment complexes that are built on a collection of parcels. The maximum of deeds for a single address in our data set was 62, which occurred in 25 specific cases in the same street where an apartment complex spanned more than a dozen parcels, each with its previous history. The mean is 1.71 deeds per address, median is 1, with a standard deviation of 2.79.

We cannot know in advance whether cases with a large date difference contain information that is of use to the machine learning model. It is, theoretically, possible for someone to have bought a parcel of land in 1992 for the purpose of constructing a house there, only to have the funds available in 2015. Let us consider the hypothetical option of limiting the set of deeds to records that fall within, say, one year on either side of the survey date. This means that dropping records outside of the one-year on either side window would dismiss 3,226 of the 15,930 (20,5%) survey records, a loss that is straining the already limited size of the data set. Dropping the records outside the window would also have a detrimental effect on a production setting. For an expected one fifth of the addresses, a prediction would not be given with any confidence since no representative data was included in the test data. Instead, we opted for a first-attempt data model T1 where no pruning on older records was done, and document term counts were combined for as many deeds as were available for a specific address. In other words: for data set T1 we expect our model to deal with the range of possible pre- and post-dated purchase deeds.

Methodology

Data set subsampling

Since we have a data set that is both class-skewed and on the small side (taking into consideration the high dimensionality of our input data), we carefully chose a stratified random sampling strategy to shaping our training and test subsets. The full data set may consist of 15.000 data points, but only 9.4% of these are the positive examples we are after. Since the model has to learn to distinguish these positives from a maximum of 1500 samples, splitting into training and test subsets becomes a matter of some consideration. A random split of 75% training data and 25% test data appears to be a popular choice: the commonly used machine learning python library *scikit-learn*[7] for example offers a subset splitting function with a default 75/25 split. However, with a 25% test set containing only approximately 325 positive examples, we run a higher risk of having a test set at our disposal that under-represents the variance in the data set. Instead, we opted for a split with a somewhat higher number of positive samples. We settled on a 70/30 split with 393 positive samples to have a better indication of model performance in the final test phase.

Secondly, we followed a stratified train/test split strategy to make sure that the training and test label distributions were identical. A 70/30 random split was made from the negative samples before applying the same ratio sampled randomly from the positive samples. The positive and negative test samples were then joined for the 30% subsample, shuffled in

[7] Pedregosa et al., 'Scikit-Learn: Machine Learning in Python'.

random order into the final test set and set apart for the testing phase. This process was repeated for the 70% training set subsample.

For the training/validation phase, we chose a repeated stratified random subsampling cross validation scheme sampled from the training set at runtime, a scheme also known as stratified Monte Carlo cross validation.[8] So, rather than persisting a fixed validation subset, the train/validate subsets are made randomly at each training run when applying the stratified Monte Carlo strategy. This has the benefit that the validation subset size can be chosen independently of the number of training runs on a specific set of hyperparameter settings, with the disadvantage that there is a chance that some samples in the set may never be evaluated in any of the runs. This disadvantage can be countered by using a large enough number of runs and a large enough validation subset size. With 10 runs with each a 30% chance of being selected, each training sample has only a 0.7^{10} = 0,0282 or 2.8% chance of not being seen in any validation set.

Through repeated training runs, we inspect the model performance variability caused by subsampling from a limited size data set and model parameter initialization. Our training procedure introduces three sources of variability: the stratified random split into train/validation samples, the random choice of data samples in mini-batches (applicable in batch stochastic and the initialization of the parameters of the algorithm. To observe the variance introduced by these sources of randomness, each training run was repeated ten-fold. The ten-fold run was chosen as it offered sufficient indication of the variance in performance results and the effect of the limited data size on the test results. The factor ten here is a rather arbitrarily chosen number, any other higher number could be chosen for a better assessment of performance variance, however we found that a ten-fold repeated run was both sufficient and cost-effective. The final test runs are trained on the combined training and validation subsets and tested on the final test set in ten-fold as well.

PDF text extraction

Model performance is greatly dependent on good data preprocessing. In the case of the purchase deeds, this preprocessing step was non-trivial. The data were supplied as Portable Document Format (PDF) files that need to be transformed from its binary format to plain text in order to be readable to common machine learning libraries for natural language processing. Extracting text from PDF is, however, a non-trivial task since a PDF file is a collection of location-specified zones containing content rather than a document of sequential text. We therefore evaluated the optimal strategy for extracting text based upon a random sample of 16 purchase deed PDF files related to the survey. For these 16 files, we created a gold standard through manual extraction by selecting, copying and pasting the textual data to a raw text file. We then selected libraries for plain text extraction for Python for comparison with our gold standard. We selected the top four[9] PDF text extraction libraries, in descending order of popularity:

1. *PyPDF2* version 1.26.0,[10]
2. *pdfminer.six* version 20181108,[11]
3. the *tika* version 1.19 Python wrapper[12] for the Apache Tika[13] text and metadata extraction framework and

[8] Dubitzky, Granzow, and Berrar, *Fundamentals of Data Mining in Genomics and Proteomics*., 178-179.

[9] By popularity, from https://hugovk.github.io/top-pypi-packages/, accessed February 4, 2019

[10] https://pypi.org/project/PyPDF2/

[11] https://pypi.org/project/pdfminer.six/

[12] https://pypi.org/project/tika/

[13] https://tika.apache.org/

4. pdftotext version 2.1.1[14] as a wrapper for the *poppler* PDF rendering library.[15]

Although popular, the following libraries were deselected:

- *pdfminer*[16] has no Python 3 support, so was dropped in favour of *pdfminer.six*;
- *pyPdf*[17] is no longer maintained and was superseded by *PyPDF2*;
- *pdfminer3K*[18] appears to be unmaintained and was dropped in favour of *pdfminer.six*;
- as was pdfminer2;[19]
- *pdfrw*[20] does not appear to support easy text extraction;
- the *textract*[21] library is covered by its pdf extraction back end dependencies *pdftotext* and *pdfminer.six* included in the selection.

For testing these libraries, we manually (visually) compared the output of the sampled PDF texts to the target texts we manually extracted and curated. From these comparisons, it became clear that visual inspection would be the best way to compare the results. This is for two main reasons: time investment and feasibility. It would be laborious to create a fully automated comparison suite, in particular due to the (often copious) use of whitespace by the different libraries. Each of the libraries inserted extra spaces or tabs in the document, and very often many line breaks as well. While it may be possible to automatically compare by discarding or contracting whitespace, one could miss evaluating situations of mid-word insertion of whitespace. This happened frequently with the *PyPDF2* library, breaking up words at seemingly random locations, disrupting the downstream tokenization process and resulting in loss of information. The second aspect is that although there are good standardized diff libraries available, these libraries have difficulty with situations of re-ordered text sections. For instance: in cases where the page number or a document identification string from a page header or footer is inserted in the wrong place in the text, the diff algorithm produces a very large diff, even though the actual change in the text is quite small. In short: direct visual assessment of differences between the gold standard and the result text produced by a PDF conversion library allows for the best indication of the performance of the produced texts.

The differences in production quality between the different text extraction libraries is surprisingly large. The effort of testing different libraries is rewarded by the conclusion that the most popular Python library actually produces the worst results on our test samples, and the least popular the best. The *PyPDF2* library often inserted line breaks mid-word, producing results that are often hardly readable. On the other end of the spectrum, the pdftotext library consistently produced the best results. It was not only able to produce texts that were nearly verbatim copies from the target texts, but it also managed to keep much of the indentation from the PDF source as well. It is therefore surprising that the *pdftotext* library scores much lower in the Python package popularity index.

Text preprocessing

When an address has multiple deeds linked, the documents are concatenated in historical order into a single text. In this stage, the document is already fit for purpose for character-level encoded recurrent neural nets,

[14] https://pypi.org/project/pdftotext/
[15] https://poppler.freedesktop.org/
[16] https://pypi.org/project/pdfminer/
[17] https://pypi.org/project/pypdf/
[18] https://pypi.org/project/pdfminer3k/
[19] https://pypi.org/project/pdfminer2/
[20] https://pypi.org/project/pdfrw
[21] https://pypi.org/project/textract/

by mapping the document characters individually to one-hot encodings of the position in the ASCII table. This creates vectors of varying length, with sequences of one-hot encoded vectors of 255 positions.

For fixed-length feature vector (shallow) models, the concatenated documents are converted to a bag of words term vector. These are produced by the following procedure:

1. The document files are converted to raw text files, tokenized, stemmed and subsequently pruned from stop words;
2. 20% of the resulting abridged corpus documents are sampled to create word counts. The vocabulary and the occurrence counts are saved into a vocabulary file, sorted descending by occurrence;
3. The top 10K most frequently occurring words from the vocabulary are selected to be included in the term vectors;
4. For each address, the words in each related document are checked for presence in the top 10K words and are marked in the term vector as present (as a binary term vector).

Model performance metrics

The model task can be described as a simple binary classification task, which has many advantages for metrics that help us evaluate model performance. A house is predicted to be either project-built ("0" or the negative samples) or self-built ("1" or positives).

The most simple method would be to measure accuracy, as the fraction of the total number of correctly predicted instances over the total number of samples in the test. Although accuracy is easy to interpret, it has a severe disadvantage which can be shown by examining the majority vote baseline. The majority vote scores 90.6% accuracy by just marking every test sample as a negative (project-built), even though it fails to retrieve a single positive (self-built) instance from the test sample. Therefore, we also use other metrics to evaluate our model performance on its ability to retrieve positive data instances. In this subsection we discuss precision, recall, and F1-score. In our model evaluation, we included both the accuracy, precision and recall.

Precision and recall

Rather than on accuracy (which gives us little information on retrieval performance), we evaluate and rate the models in this investigation along the lines of precision and recall. The intuition behind precision and recall is as follows: precision is the fraction of correctly identified positives (true positives) in the predicted positives (true positives and false positives), i.e. the portion of the predicted positives that the model actually got right:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Recall is the fraction of the positives in the test data that the model was able to retrieve:

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}.$$

As stated earlier, we can allow for some leeway in model precision, if this results in a higher recall. In order to set decision threshold between a predicted positive or negative, we set our classification models to be ranking classifiers, i.e. models that produce a scalar value indicating a probability or confidence level leaning towards either a positive or a negative prediction.

F1-score

Since we aim to maximize the recall on positive, self-built instances in our data, we also opt for a F1-score (or f-measure):

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall} + \epsilon}$$

Our definition differs slightly in that we add a small epsilon for computational stability, in order to avoid dividing by zero if we have no retrieved true positives. With the addition of the epsilon in the denominator, we can obtain an F1-score of 0 even when there are no retrieved true positives. Like the accuracy metric, the F1-score ranges from 0 to 1, where in both metrics a score of 1 equals a test where every test sample is classified correctly. The difference is in the F1-score penalty on both precision and recall. While a simple accuracy score on the majority vote baseline can attain a score of about 0.9, the F1-score for the majority vote is 0, since both the precision and the recall are 0.

The advantage of F1-scores is that it provides a single metric that is more informative than an accuracy score. However, in many cases, as is the case even with F1-scores, statistical performance metrics also obscure a simple view on the performance of models.[22] For instance, there are at least four[23] methods of calculating an F1 score in binary classification:

1. **'binary'**: assume instances labeled '1' or 1 (i.e. as string or integer) as positive class and calculate the score for these instances only. This equates to the F1-score method given at the start of this paragraph;
2. **class-agnostic**: by calculating the scores for all classes;
3. **macro**: by calculating metrics for each label class by unweighted mean, disregarding label imbalance.
4. **micro**: calculating score by counting the total true positives, false negatives and false positives.

Although there may be arguments for any of these variants, we chose the default 'binary' option in *scikit-learn* as our metric, since we also have a binary classification problem where a minority class of positives needs to be detected. A second problem of F1-scores is that the different aspects of precision and recall for our experiments remain hidden. This is why we chose to list model precision and recall marks in our results table for a more fine-grained result.

Data set size evaluation

In this subsection we consider the methodology of inspecting the problem of the size and label distribution of our data set. With 14k instances in our full data set and a sizeable class imbalance of about 10% positive instances, we need to evaluate whether our data set is actually large enough for our supervised models to be fit for purpose on our range of models. This a common problem known from many fields working with predictive models and limited data.[24] The problem of limited data set size is aggravated by the data set size requirements that grow with the dimensionality of the data, known as the *curse of dimensionality*. Given sufficient spread of variance over the dimensionality and individually distributed features (orthogonality), the higher the dimensionality of the data the more data instances are needed

[22] Demšar, 'On the Appropriateness of Statistical Tests in Machine Learning'.
[23] See for example https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score and more in-depth at https://datascience.stackexchange.com/questions/15989/micro-average-vs-macro-average-performance-in-a-multiclass-classification-settin
[24] Figueroa et al., 'Predicting Sample Size Required for Classification Performance'.

for the model to learn a mapping without overfitting.

The starting point is that we have a few default baselines such as majority vote, minority vote and random (Bernoulli) choice, combined with metrics such as accuracy, F1-score and Jaccard index. Our starting point is to assume that our models trained on minimal exposure to our data will mimic one of these starting point baselines. After training exhaustively on just a few samples from the training set, our models are almost guaranteed to overfit on these few instances and therefore achieve a perfect fit on the training samples. The generalised model performance, however, will be very poor, as evidenced by the performance on the prediction on the validation set. The expected result on the validation set will therefore be an accuracy score of about 0.9 and a F1-score of about 0.

However, we expect our model to learn something more useful from exposure to more training data. We allow it to learn a mapping that is more sophisticated than achieved by overfitting to a very small subset of training instances. In short, the more training data we allow our models to learn from, the smaller the expected gap between training and validation performance metrics. In an idealized situation, we hope to see the model validation performance grow from an F1 score of 0, asymptotically flattening to a score of 1. A more realistic result, however, will be of a erratically increasing validation performance curve from 0 to somewhere between 0 and 1. Given enough data, the performance will converge to a performance cap depending on a number of factors:

- the model algorithm may or may not be able to model the desired function of a full 100% accurate mapping from the input data to the labels;
- the configuration settings (hyperparameters) for the model may be well or poorly tuned to the subselection;
- the selection of input data features may or may not be informative enough to allow the algorithm to create an accurate mapping;
- the sample selection may or may not be large enough to cover the variance in the pattern that can hypothetically be discovered in the full data set.

Distinguishing between the roles of these different factors is a matter of experimenting with different input feature configurations, evaluate different model types, and do repeated runs with stratified Monte-Carlo cross validation subsampling with per-run hyperparameter optimization.

Using this principle of expected performance convergence, we can experimentally evaluate the question on whether our data set contains enough data for use in a particular combination of model and feature selection set. From this evaluation we can determine whether it is useful to re-train the model using (much) more data. If the gap between training and validation score is bridged by adding more data, then there is no immediate need to collect more data. However, if a sizeable gap remains while the performance curves show no sign of converging (i.e. have a notable positive gradient for the validation performance curve and a negative gradient for the training performance curve), it strongly suggests that collecting more data is likely to improve model performance.

Model evaluation

In this project we have data from two different modalities: linguistic information in the form of texts, and symbolic/descriptive information in the form of general descriptive properties of the address. We designed our experiments to

evaluate the added value of each of these modalities.

First, we rank the modalities for each learning model type descending from most informative to least informative. We do this by hyper-parameter tuning each model in order to obtain the best F1-score of each model, and we include the test results of these runs in Table 1. These are considered to be our baseline models.

Results

Since we have a large set of model test results, we have included a table of results (Table 1), for which we discuss the individual models performance interpretation.

Results on attribute feature set

We investigated the question to what extent the size of the data set limits the accuracy of our models, and to what extent the data can be fitted on spatio-temporal features. The data for this model, feature set A, consists of data that is already part of the survey data itself: the contract date, postal code and house number specifics (house number, house letter and house number addition). The purpose of this particular set of features is to expose temporal (date of sale) and spatial (address) bias in the data set.
We attempted a logistic regression model at first, but it was unable to significantly exceed a majority vote score. The performance was poor enough not to include the results in the table. As a second experiment we trained a baseline model using a decision tree, which performed much better, reaching an 10-fold randomly stratified subsample average accuracy of 0.960 at a standard deviation of 0.00224. This is almost 6% points above the majority vote. The implication of this is that there is considerable temporal and spatial bias in the data that may be reflected to a larger or lesser extent in the other features, such as the deed texts. For example: chances are that a certain notary firm, indicated by name in a text, operates in a certain geographical area that may or may not have a higher percentage of self-built dwellings.

Besides obtaining a ten-fold validation accuracy average, we investigated the model performance on random stratified subsamples of increasing size from 1 to 100% of the training set, at a step size increase of 1%, resulting in 100 consecutive runs. The results of these 100 runs shown in Figure 4 allow us to deduce several things:

1. The training accuracy and F1-score performance are almost equal to 1 across all subsample sizes. This means that the decision tree was able to fit the data perfectly in almost every random subsampled set, regardless of its size.
2. The validation accuracy and F1-score increase on larger subsample sizes. This is understandable: the decision tree algorithm is able to create a better

Model type	*accuracy*	*precision*	*recall*	*binary F1-score*
Majority vote	0.906	0	0	0
Minority vote	0.094	0.094	0.094	0.171
Decision tree A1	0.960	0.845	0.697	0.764
Logistic regression T1	**0.985**	**0.934**	**0.899**	**0.916**

Table 1. Table of baseline scores and model results for distinguishing self-built homes from project-built homes. Data set A1 based on attribute data, data set T1 based on textual data from

mapping from input data to labels using more data. The validation binary F1-score reaches 0.770 at 100% of the training data.

3. The flattening curve at the 90-100% subsample range boosted our confidence that we have enough data to conduct other experiments on the data set. For the particular task (self-built detection) on this feature subset (A1) using this model (Decision tree) a larger training set will not likely constitute a large increase in F1-score performance.

Results on text feature set

After simple spatio-temporal features, we tested a simple logistic regression model on data set T. T constitutes a data set of simple binary term vectors (unigrams) of the 10,000 most frequently occurring words. The T vectors were based on a corpus of stemmed words, where common Dutch stop words were removed.

To compare the data set size requirement with the earlier A-experiments, we also ran a series of training sessions of increasing size in steps of 1%, from 1% to 100%. The resulting plot in Figure 5 shows that, for convergence on a stable performance score, logistic regression on data set T requires a lot less data than the decision tree required for data set A. The model converges to good scores at already about 30-40% available training data.

For a better grasp of model performance, we conducted 10-fold stratified Monte-carlo cross validation runs. Over the 10 runs, the model scored an average validation accuracy of 0.985, with a quite low standard deviation of 0.00145. The only model hyperparameter that was tuned on the size experiments and 10-fold

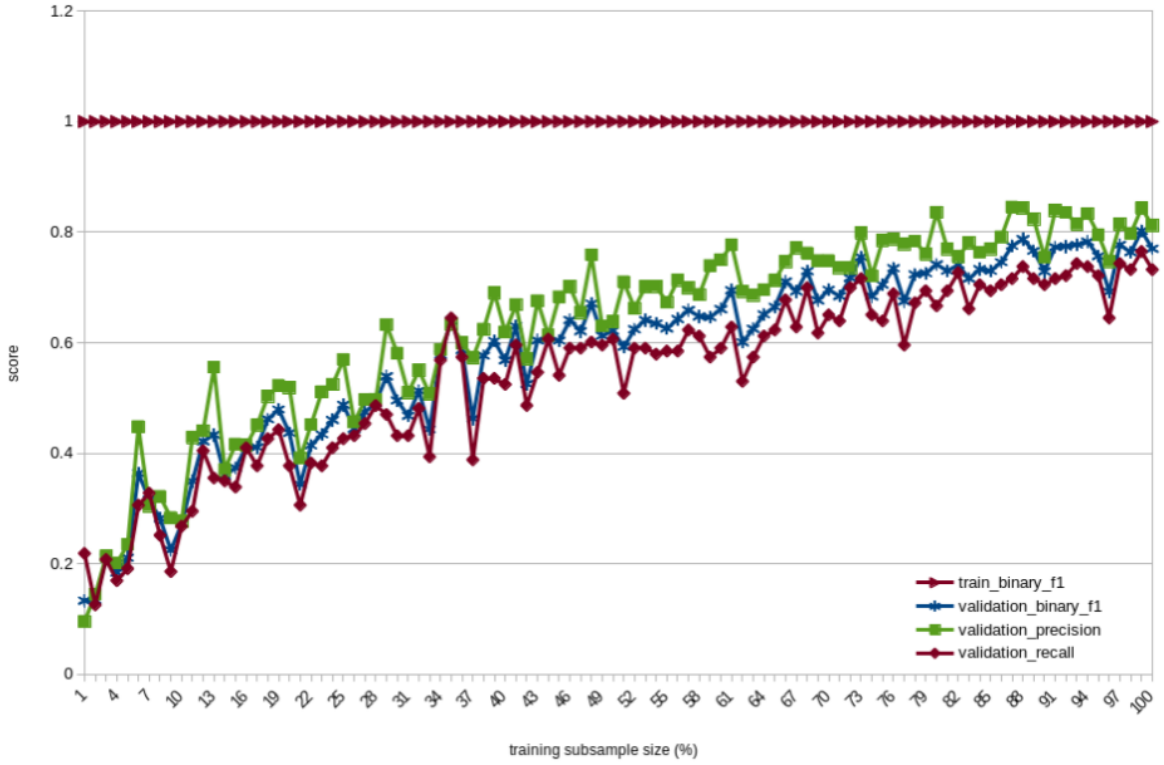

Figure 4: Decision tree (unconstrained depth) performance on feature set A1 (spatio-temporal data). On the horizontal axis the runs on subsamples of increasing size in percentage of the total available training data.

cross validation runs, was that of the maximum of iterations allowed: this was increased to a maximum of 512 iterations. The model was trained on a limited-memory Broyden–Fletcher–Goldfarb–Shanno (*LBFGS*) solver[25] with l2 regularization and an early stopping tolerance of 1e-4. With the exception of the maximum number of iterations, these are all default settings for the chosen implementation of logistic regression in scikit-learn.

Model interpretation

To know what the model has learned, we inspect the model intercept term (bias) and the coefficient distribution over ten runs. The best performing models produce a negative intercept term, which is logical given the class imbalance. From repeated experiments it became apparent that there is considerable spread of intercept term size and coefficient distribution, meaning that there are several possible local optima that the model is able to reach, producing similar accuracies. The model always has a negative intercept term of mean -1.3 at a large standard deviation of 0.50. This large spread makes it a little bit more difficult to interpret 'what the model does', since each run has a different outcome depending on random model coefficient initialization and random Monte Carlo cross validation splitting.

In order to get a reading on the model workings, the model coefficients were averaged over ten training runs. The coefficients are distributed close to a normal distribution, with a mean of -0.001 and a standard deviation of 0,0723, indicating that

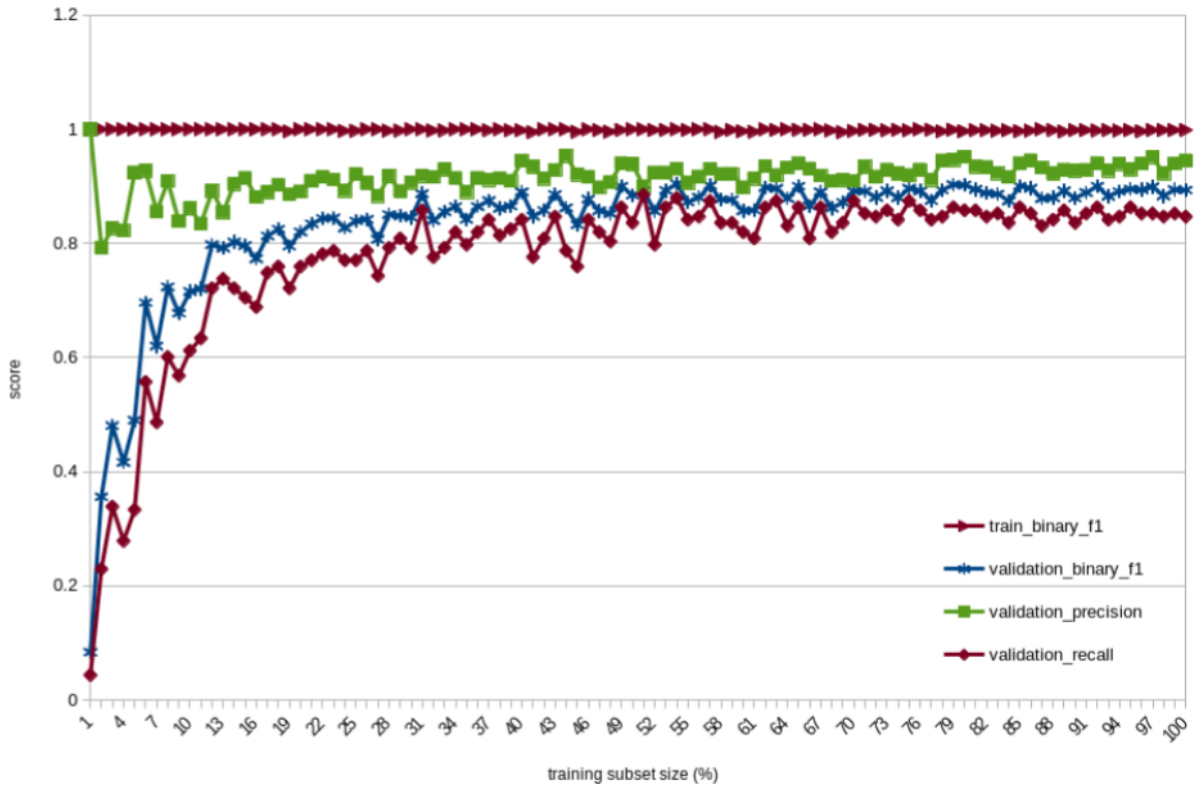

Figure 5: Logistic regression performance on feature set T1 (deed texts). On the horizontal axis the runs on subsamples of increasing size in percentage of the total available training data.

there are very few terms that have a large influence and many that have nearly no

[25] Fletcher, *Practical Methods of Optimization*.

influence (see Figure 6). There is no single word in the top 10,000 corpus words that creates a positive (self-built) prediction; it is always a combination of terms that would create a positive prediction. In the next paragraph we list the top-10 most positive and negative influencing words. In this post-hoc analysis we find for most words an obvious reason for influencing positively or negatively. For some words this is much less obvious, although their influence is significant.

Some of the most positive, self-built influencing words are, in descending order:

1. *heer*, stemmed from *heren* or *heerder* or *heerend* or *heerde* or *heer*, meaning 'mister' (with a high occurrence rank: 477), indicating a private individual rather than a company.
2. *huw* (occurrence rank: 981) as stemmed from *'gehuwd'* or 'married', which is a strong indicator that the buying party mentioned in the deed is an individual or a married couple, rather than a development company.
3. *kavelnummer* (477), the identification number of a lot, in contrary to a "bouwnummer" which is the identification of a to be realized building (nr 2 negative influencers);
4. *onverdeel* (824) stemmed from *onverdeeld* or "undivided", a legal expression indicating private buyers;
5. *5* (102), most probably from "artikel 5", defining guarantees in agreement of the transaction. No obvious reason for positive influencing.
6. *tezaam* (321), stemmed from *tezamen* or "together", no obvious reason for positive influencing.
7. *helf* (830), stemmed from *helft* or 'half'. Indicating a split in ownership with private persons involved in the transaction.
8. *09:00* (694). Deeds that are offered for registration in the "Openbare

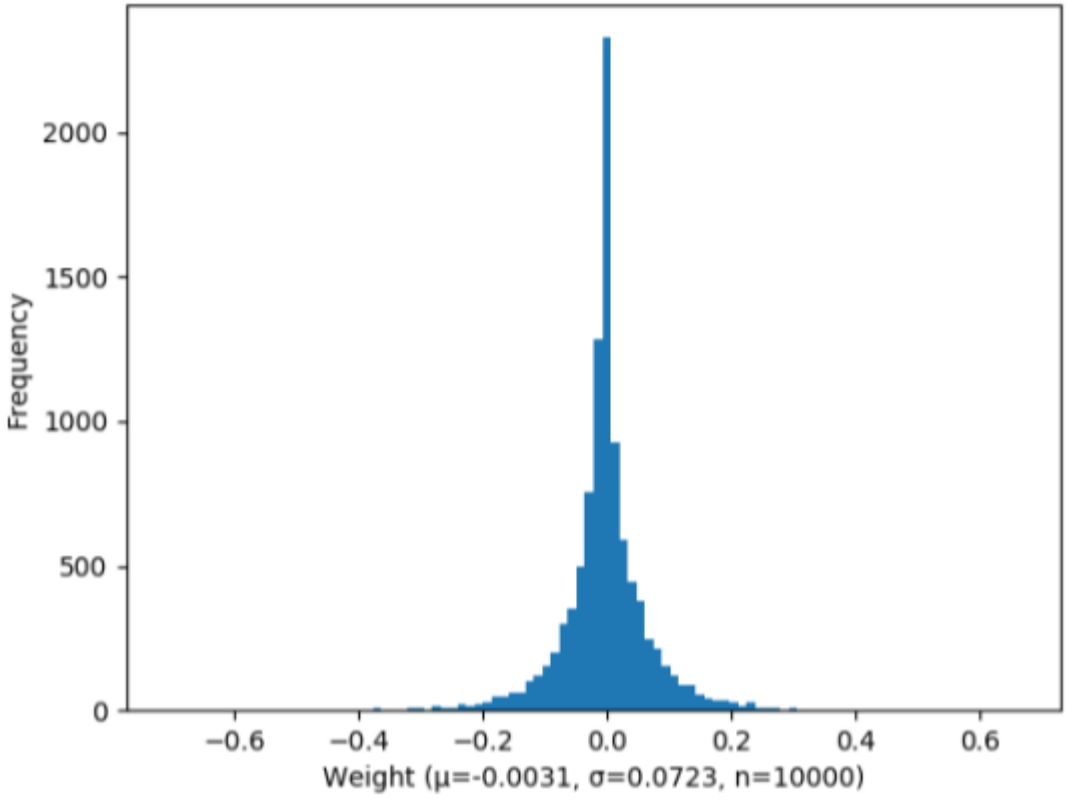

Figure 6. Average weight (model coefficient) distribution histogram over logistic regression model of the top K=10,000 most occurring words in the T1 corpus. The 10,000 model coefficients are aggregated into 100 bins.

Register" at the "Kadaster" on working days after 15:00 are processed with the registration date the following workday at 09:00. Deeds with a selfbuilt label are more likely to receive this "09:00" stamp, which should be interpreted as a proxy for other reasons such as the working routine of a notary firm that processed a lot of selfbuilt homes.

9. *vrijstaand* (3394), indicating a detached building. This is logical, since most self-built homes are detached.
10. *waarover* (4621) or "over which". No obvious reason for positive influencing.

The top most negative influencing words in ascending order (because of negative weights) are:

1. *aannemingsovereenkomst* (occurrence rank: 363), stemmed from *aannemingsovereenkomsten* or *aannemingsovereenkomst*: meaning 'agreement of acceptance'. This agreement is signed mostly in cases where there is a project built dwelling on the land parcel that can be agreed upon its acceptance by a buyer.
2. *bouwnummer* (134) or 'construction number'. This word indicates a temporary lot number before actual addresses are assigned, mostly on larger scale development projects.
3. *aanneemsom* (749) or "contract price". This is typically for project building, as it is the price agreed upon for constructing the dwelling.
4. *koop-/aannemingsovereenkomst* (599), similar to 'aannemingsovereenkomst'.
5. *besloot* (323) stemmed from 'besloten' or 'closed' . This is the first term of 'besloten vennootschap', or 'private company', indicating a company ownership rather than private individual.
6. *opleveer* (374), stemmed from 'deliver' or 'delivery'. This indicates a project wise building project, where seller and buyer agree on a specific delivery date.
7. *woningborg* (1189) or 'dwelling deposit', this is an insurance policy on project-built houses.
8. *statuair* (485) or "statuary". The significance is not clear.
9. *elf* (551) or "eleven". No obvious reason for negative influencing.
10. *project* (490), The significance is not obvious; it might occur when the name of the building project is mentioned in the deed.

Results on the test set

To confirm the model performance we conducted runs on data the model has never seen before; the test set for A1 and the test set for T1 we subsampled in the data set subsampling chapter. We expected the model to perform similar as on the training data.

Model	**A1 Decision Tree**		**T1 Logistic**	
Dataset	**Training**	**Test**	**Training**	**Test**
Accuracy	0.96	0.96	0.99	0.98
F1-Binary	0.76	0.79	0.92	0.92

Comparing the scores it is satisfying to see this turns out to be the case.

Conclusion

The purpose of this feasibility study was to investigate the possibility to create a 'self-built home detector'. Backed by four years of data gathering on newly-built homes by the CBS on the one hand and the national registry on land parcel transactions at the Cadastre on the other, the two institutions partnered up to cooperate in this study.

The feasibility study operated from the premise of relatively simple data mining and data science techniques, to switch only to more complex methods if the performance of simpler methods were to be insufficient. Two different data sets (spatio-temporal data and textual data) were tested, using two different machine learning models (decision trees and logistic regression).

Using this two by two experiment, this proved enough to show that detecting self-built homes is indeed a clear possibility. The text mining approach using a simple logistic regression model showed promising performance on retrieving the self-built cases from the validation set.

Potential improvements

Since it seems possible to realize an additional breakdown for transacted newly built dwellings into self-built dwellings and dwellings built by developers, this opens the door for several potential improvements in the area of real estate statistics:

1. Improvement of the weights of the OOHPI.
2. Improvement of the newly built dwellings in the HSI.
3. Explore the possibilities for statistics about self-built dwellings, since this was never possible before.
4. Use this technique to create other classifications of dwellings which do not exist yet.

Future work

To actually translate the results of this feasibility study into a production system, a lot of work is still to be done. A few issues can be identified already.

Since the objective of CBS is to eliminate the minority class of self-built cases from the newly-built dwellings dataset, we can opt for a strategy with less precision, but higher recall. In short: we can allow the model to misclassify some cases of project-built samples as self-built and remove them, but only to a certain extent. Future work should reveal the optimal balance of the strategy.

A second issue is that it is still an assumption that these deeds can offer better and more reliable information than, say, data from aerial photography, topographical data and neighbourhood statistics. Even if deeds provide the best information source, it is still unknown whether model performance improves even more when different data sources are combined. Future research therefore should include different scenarios of trying single data sources and combined data sources in order to explore the hypothesis space of data value.

Third, some improvements on the modelling are possible. The current model is based on single words, ignoring word combinations and sentence structures. Adding that information to the model might improve its predictive power. In addition, currently the model is based on binary term vectors for the 10K most frequently occurring words in the dataset. Giving the strong class imbalance in this dataset, this probably lowers the chance of typical self-built words to be included in the

vocabulary, giving the model less power to identify self-built dwellings. For the future a better feature selection is proposed, for example with the help of a principal component analysis.

Fourth, implementing the proposed methodology into the current production process raises questions on the practical feasibility. Amongst these questions are:

1. To which time extent is the Cadastre able to collect (a certain minimum threshold of) the needed deeds, mine them, run the model and add the resulting classification to the currently existing quarterly dataset delivery of sold newly built dwellings to CBS? And and to what extent is it possible to link up with the already existing regularly delivery of data of newly built homes?
2. Does this concept raise any problems on privacy compliance for both CBS and Kadaster?
3. Does this require adaptations to the current production time schedule of both organizations?
4. Which way of adding the resulting classification to the current data delivery is feasible for both Cadaster to deliver and CBS to process?
5. Is CBS able to implement the new classification in the production process of OOH and HSI? Does this lead to new, unforeseen problems?

Future work is required to answer these questions and give definite conclusions about the overall feasibility of this methodology to exclude self-built dwellings.

Bibliography

CBS. *Prijsindex Nieuwbouw Koopwoningen. Methodebeschrijving*. Centraal Bureau voor de Statistiek, 2018. https://www.cbs.nl/-/media/_pdf/2018/28/2018ep34%20methodebeschrijving%20pnk.pdf.

Demšar, Janez. 'On the Appropriateness of Statistical Tests in Machine Learning'. In *Workshop on Evaluation Methods for Machine Learning in Conjunction with ICML*, 65, 2008.

Dubitzky, Werner, Martin Granzow, and Daniel P Berrar. *Fundamentals of Data Mining in Genomics and Proteomics*. Springer Science & Business Media, 2007.

Eurostat. *Technical manual on Owner-Occupied Housing and House Price Indices*, 2017. https://ec.europa.eu/eurostat/documents/7590317/0/Technical-Manual-OOH-HPI-2017/, viewed on June 17, 2019

Figueroa, Rosa, Qing Zeng-Treitler, Sasikiran Kandula, and Long H Ngo. 'Predicting Sample Size Required for Classification Performance'. *BMC Medical Informatics and Decision Making* 12 (2012): 8. https://doi.org/10.1186/1472-6947-12-8.

Fletcher, Roger. *Practical Methods of Optimization*. John Wiley & Sons, 2013.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 'Scikit-Learn: Machine Learning in Python'. *Journal of Machine Learning Research* 12 (2011): 2825–2830.