Technical documentation

WPI model onderzoekswaardigheid uitkeringsaanvragen

Developers	
Model version	1.0.0
Date	August 3rd, 2022

1. Introduction

The department Werk, Participatie & Inkomen (WPI) is responsible for welfare benefits in the gemeente ("uitkeringen"). This project focuses on the application process for those uitkeringen. More specifically, we aim to provide a risk score for incoming applications, that can help to determine if an application should be investigated in detail, or only screened shortly.

More background information can be found in the non-technical documentation.

2. Terminology & working process

WPI is the department with whom we are doing the project.

Uitkeringen are welfare benefits. Although WPI is responsible for many types, the scope of this project is regular uitkeringen for people aged 27+. The person receiving the uitkering is often called **klant**.

The **inkomensconsulent** (**IC**) is currently the first person who looks at an application for an uitkering and is responsible for granting (**toekennen**) or rejecting (**afwijzen**) the application. To do so, they carry out a **screening**, which takes approximately 2-3 hours.

If an application looks suspicious to the ICer for whatever reason, they can scale up to a **handhavingsspecialist/handhaver** (**HH**). This person carries out a more thorough investigation (**handhavingsonderzoek**) that takes around two days and sometimes includes a house visit. After their investigation, the handhaver gives an advice to the inkomensconsulent about whether to reject the application (**afwijzen**)**,** grant it without changes (**geen wijziging)** or grant it with changes (**wijziging**). The inkomensconsulent then makes the final decision, which is generally in line with the advice, but it may differ. In the data, an investigation by handhaving is called a **proces**.

The **klantmanager** is the person who manages contact with the klant for the duration of their uitkering and aims to help them find work or education. This role is not as relevant for the project as the other two.

A **product** is a type of benefit that the municipality offers. Some of those are paid out in intervals for a longer period, whereas others are a one-off payment. Each product has a product number that identifies it. An instance of a specific product received (or applied for) by a specific person during a specific period of time is called a **dienst**.

Many of the decisions in this document were made based on input from the **werkgroep**, a selection of handhavers and inkomensconsulenten (and sometimes others) who helped with the interpretation of data and generally provided a lot of immensely useful insights about their work.

3. Source systems

We get data from three systems in use at WPI.

1. **Socrates**: Used (mainly) by the inkomensconsulenten. It contains information about people, their uitkeringen/applications, relations to other people, income, address, etc.
2. **Sherlock**: Used by the handhavers. It contains information about the onderzoeken.
3. **RAAK**: Used by the klantmanagers. It contains information about contact with klanten, their work experience, education, etc.

The sensitive data that could be hashed without losing important information for the model, has been hashed using SHA256 and a salt value. Since for the correct functioning of the model we need the same start-values to be mapped to the same hashed value, instead of using the salt with the password, we concatenate it to the value itself. This allows to resist dictionary-based attacks and keep the consistency.

The data used for model training has been hashed in the VAO and uploaded to Azure. The production data is being hashed by team DASO and provided to the model through an API. What exact data is hashed is described in the WPD.

4. Dataset

4.1. Scope

Applications are taken from the Socrates table Dienst. Included in the dataset are applications made between January 1st, 2015 and November 2nd, 2020. The end date is limited by the provided data dump. The start date of the dataset was decided together with the business; it's within the legal retention period, provided us with a reasonable amount of data for development, and the data is assumed to be similar enough to applications nowadays. Like humans, data grows older by the day, so setting a hard limit on how long we can look back is difficult. Therefore, we agreed to handle this pragmatically: The pilot is done with the model based on the dataset as described, and this dataset will be updated when the model next gets retrained, so that it includes newer data and excludes data that has gotten too old (more than 5 years ago).

Furthermore, we only consider applications for the relevant (as determined by the business) product numbers, which are 131 (Wwb/LO), 135 (Wwb/EV), and 227 (krediethypotheek). Other product numbers are excluded, because it's important that the products are similar enough, so that the same legislation applies and by extension the same justification for (not) using certain features.

Also excluded are applications from special groups (**Bijzondere Doelgroepen**), such as homeless people, because they are handled by separate teams. Most can be identified by their address, because

many of them are registered in the system on a municipality address[1]. If they've been investigated by HH before, the team that investigated them is another indicator.

The corona period is included in the dataset after two considerations. First, products that only existed during the corona period are automatically excluded since their product number is not selected. Second, for the rest of the data, we performed an analysis to understand if the data in this period behaved substantially differently, concluding that this is not the case.

The applications that remain after these filters can be divided into three groups based on how they were handled:

1. Applications that were only screened by IC (we will refer to those simply as 'screenings').
2. Applications that were screened by IC and subsequently investigated by HH (we will refer to those simply as 'investigations').
3. Applications that were only screened by HH; this is a special category that occurred during the corona period, because the department was flooded with applications and handhaving stepped in to help with the screenings (we will refer to those as 'screenings by handhaving').

Most applications fall into group 1, because the capacity for investigations is limited, so group 2 is not very big. In the end, we decided to limit the dataset only to group 2, so the applications investigated by HH. This unfortunately reduced the size of the dataset by a lot, but it proved difficult to determine a reliable enough label for applications that were only screened – see for more details the section 'Label'.

This reduces the representativeness of the dataset, in the sense that we expect (and saw this confirmed in the pre-pilot) that the model will produce higher scores on average on the train datasets than in production. This, because the train dataset contains only investigations, and investigated applications are, by definition, more 'suspicious' than the average application. However, we can correct for this by adjusting the classification threshold. Moreover, we think that the negative impact of having shaky labels is worse than the impact of using a smaller/slightly less representative dataset.

4.2. Content

The BI team of WPI provided us with 28 tables from Socrates, 41 tables from Sherlock and 9 tables from RAAK. These are not all the tables that exist in those systems, nor were all of them incorporated in the model. The file *Overzicht Verwerkte Data en Features.xlsx* provides an overview of the data that *was* incorporated. An entity-relationship diagram of (the majority of) the tables, created by team Advanced Analytics, can be found on the Sharepoint.

Some of the tables contain numerical codes representing strings. The meaning of those codes can be found in one of the reference tables (Socrates) or sys tables (Sherlock).

4.3. Data quality

We first analyzed the quality of the raw data, and then did some more sanity checks (min/max values, outliers, missing values) on the features (see Section 5). A few issues are worth mentioning:

[1] The exact addresses were specified by Handhavingsspecialisten. We also reverse checked some addresses with them that we found in the data with suspiciously many people.

- In the working process, definitions and work instructions are not always clear, sometimes leading to inconsistent interpretations and use of columns, categories, etcetera. We handled this by including a broad (in number and in expertise) group of people in our efforts to understand the data. In some cases, this led to not using a column.
- Some columns contain old values that are not possible anymore due to changes in the working process or the law, for example types of uitkeringen that don't exist anymore. Where relevant, we discussed with the business if those old values should be treated the same as some other values, or if we should exclude them completely.
- Some of the systems keep a history, which shows that sometimes mistakes are made while entering data that get corrected shortly after. We're able to filter those mistakes out.
- The formatting of data is not consistent - not between systems, not between columns within the same system, and not even always within a column. This goes in particular for date columns, but also for some categorical columns, for example when the same category is present both with and without a full stop at the end. Such issues were fixed manually and automatically during the ingestion of the data.
- Some columns are abused to register other information than what the columns is for. This is not an issue, because they are all free text fields that we don't use anyway for privacy reasons.
- Some data that exists in one system does not exist in another, although that would be expected. This is not on a large scale.

Other points of attention such as completeness and duplication show no serious problems. Topics such as representativeness and biases were analyzed in a bias analysis, the results of which are described separately in the bias analysis documentation.

It's possible that a small number of data issues still remains in the dataset. Machine learning focuses on learning general patterns in data that are also valid for the future. Rare data does not substantially influence those kinds of general patterns. Hence, we expect that this does not pose a problem for the model, because we assume the number of remaining issues is limited.

In general, it's good to note that no formal documentation on the data is available, meaning we were highly dependent on the knowledge of colleagues in the organization to interpret the data correctly. The project's Sharepoint contains an Excel *datavragen.xlsx* with questions about the data (interpretation of fields, missing values, etc.) and the answers we gathered.

4.4. Other notes

Subjectnr

Note that the subject number can refer to a person, but also to an organization or to a partij (see below). For instance, organizations offering help/work/education to people that are on welfare.

Partij vs. persoon

Eligibility for welfare depends not just on the applicant themselves, but also on their partner or whoever else they may share costs with. This information who shares costs with whom ('gezamenlijke huishouding') is also registered in Socrates and it's called a **partij.** Everyone is part of a partij, so even if you're not sharing costs with anyone, you'll be in Socrates as part of a 1-person partij. A partij thus consists of 1 main client ('**hoofdklant**'), and potentially one other person, whom we'll call the

secondary client. The main client is the person who applied for the welfare. The table Partij tells you which people are in which partij.

In the table Dienst, diensten can be looked up only by subject number of the main client. In the table Dienstsubject, diensten can be looked up for the partij and for the main client. Therefore, be aware that in Dienstsubject, the subject number is not that of a person, but that of a partij.

This also means that if we want to get a complete picture of all the diensten that a person is receiving, either as main or as secondary client, we need to combine Dienst (to get the diensten where the person is the main client) and Dienstsubject (to get the diensten where the person is the secondary client).

5. Features

5.1. General

Features are based mostly on the information in Socrates and RAAK, and some Sherlock. They are partly inspired by indicators that the werkgroep told us, and partly by what was possible with the data we were given. An overview of all the features and the data that feeds into them is part of the documentation (*Overzicht Verwerkte Data en Features.xlsx*).

5.2. Filtering relevant rows

Depending on the type, a feature is calculated either as a snapshot at the time of the application (e.g., number of active addresses), or looking back over the year preceding the application (e.g. percentage of 'deelnames' started over the past year). We go back at most one year for two reasons: First, so that the information is not too outdated at the time of the application. Second, out of ethical considerations; the past should at some point be the past. We checked the effect of looking back two or even three years, but that didn't improve performance or even reduced it.

All the tables in Socrates, and some others, keep a history of facts by keeping old rows in the database, even as rows with newer information are added and displayed in the application. This means that to create the dataset, we must filter the rows on the information that was applicable, valid, and known at the time of the application:

- Applicable means the most current entry or everything not older than 1 year.
- Valid means the row should not have been invalidated.
- Known means it should have been present in the system at the time of interest.

Keep in mind that for a representative dataset, the time of interest is the application date (at least roughly, see below), so all three criteria should be checked relative to that, not to today's date. We call this the reference date.

Our experiments suggest that the closer you get to the decision date, the better the model performs, presumably because more and/or better information is available, because an IC or HH has already worked on the application. We tested that by comparing the performance when using (in order from early to later in the process) the application date (*dtaanvraag*), first entry date (*first_dtopvoer*),

investigation start date (*pro_startdatum*), and final decision date (*dtopvoer*). Currently, the model runs at most one day after an application enters the system, therefore the reference date is *first_dtopvoer*. It doesn't make sense to run the model later in the process, because its goal is to sort work between the IC and HH *before* anyone has checked the application.

5.3. Outliers

There are outliers present in the features. The most prominent is in the year of birth, the oldest being the year 1800. We decided to replace everything before 1905 by the average year of birth in the dataset. This cutoff corresponds to the birth year of the oldest person ever in Amsterdam.

Other outliers that we identified were:

- *total_vermogen* had high negative and positive values
- *application_count_last_year* had a maximum value of 33
- *relocation_count_last_year* had a maximum value of 18
- *total_inkomen_netto* had a maximum value of 100K

None of those are 100% impossible, only improbable. Combining this with the fact that we use a tree-based algorithm, we decided to leave them be, since this algorithm type should be able to deal with outliers natively.

5.4. Missing values

We deal with missing feature values in various ways, depending on the meaning of the feature and the cause of missingness:

- Some features are imputed using a high value that wouldn't normally occur. This is the case for features counting the number of days since some event happened. Since the input dataframe gets filtered on a period of X days before the application, a missing value means that the event either never happened for the applicant or it happened longer than X days ago. Hence, imputing with a value higher than X makes sense.
- Some features are imputed with zero, because missingness means that the indicator should be false.
- Some features are imputed with the average or the median, assuming this will have the least impact on the model. The decision between average and median was taken by looking at the skewness of the distribution and, in some cases, trying both.

Additionally, we create indicators of having been imputed for some features, if we think that the information being missing is meaningful in itself.

5.5. Encoding categoricals

Note that the final selection of features of the current model does not contain any categorical features, but we did experiment with how to process them best while developing the model.

Our categorical data is generally not ordinal, which means it needs to be encoded in some way to be used by the model. Two particularities make this challenging. First, for some features there are categories with very few observations. Second, for some of the categorical data one person/application can belong to multiple categories, for example if one person had multiple appointments of different types in the past year. It was a challenge to feed this information to the model properly. In some cases, we experimented with a "summary string" of ones and zeros to indicate which categories were "active". The downside is of course that this creates even more categories with even fewer observations per category. In most cases, we ended up processing the information with human knowledge, for instance by grouping categories logically or manually creating indicators that matched a business rule.

The features that we kept as categorical are encoded using embeddings. A very basic neural network consisting of 1 embedding layer is trained directly on each categorical feature, with the label as target. The embeddings can then be used at predict time to position each category in a low-dimensional space that relates to the target values and to the other categories. The number of neurons in the embedding layer is dependent on the number of unique categories of the features: it's the square root of the number of unique categories, rounded to the nearest integer. We found the embeddings to work best after experimenting with multiple alternatives, including:

- One-hot/multiple-hot encoding: create an indicator column per category. This idea was dismissed for most features due to the large number of categories, which would cause the number of columns to explode.
- Target encoding: replacing a category with the mean target value for that category. For the categories with very few observations we instead used the overall mean.
- Bayesian target encoding: same as target encoding, but then with the mean implemented as the prior, which loses weight when there are more observations in a category. Therefore, the more observations, the more the encoding matches the mean of the category rather than the overall mean.

Note that, if in a future round of development categorical features are included in the model, the question of interpretability deserves special attention, because embeddings are not naturally interpretable and it's not a given that explanations like SHAP or EBM work well for them.

6. Label

6.1. Definition

Goal of the model

Defining the label is not straightforward. To assist with dividing incoming applications between IC and HH, the goal of the model is to identify **onderzoekswaardigheid**, i.e. the extent to which an investigation of the application would be beneficial. The idea is that those applications with the highest onderzoekswaardigheid should be sent to HH. Hence, the meaning of the label is:

- 1 = 'onderzoekswaardig'
- 0 = not 'onderzoekswaardig'

It's good to mention explicitly that this is something different than identifying which applications should be rejected or granted (with or without changes). Likewise, it's different than training a model to mimic the decisions of IC to scale up to HH, in which case you're unlikely to identify applications

that should be scaled up but weren't in the past. These two points are important, because both these labels could be based directly on the historical outcome/treatment of applications, whereas the onderzoekswaardigheid is something we have to deduce ourselves.

Further complications come from the fact that we cannot be sure that the decisions made in the past were correct. This is the case especially for screenings, because they are shorter, and even more so for applications that were granted, because while rejections must be substantiated very well and may even be challenged in court, wrongful grants don't get weeded out the same way.

HH investigations

According to the werkgroep, an investigation is useful if it results in an advice to reject the application or grant it with changes, so those cases are labeled as onderzoekswaardig. We expect that this label is fairly reliable, although there may be some pollution with applications that were very easy to decide on. For instance, if someone does not live in Amsterdam, they are never eligible for benefits here. Those applications should not be scaled up to HH, but if HH still gave an advice, which in most cases they do, we cannot tell them apart.

IC screenings

This section describes the considerations around a label for the IC screenings. We eventually decided to exclude IC screenings from the dataset, because we did not trust the label enough, so you may also skip this section.

For IC screenings, the label is unclear. We considered three alternatives:

1. Label all screenings negative.
2. Label screenings with a rejection outcome as positive.
3. Label all screenings negative for the training dataset (as in point 1 above), but evaluate performance on a different label where screenings with a rejection outcome are labeled positive (as in point 2 above).

The logic of option 1 is that if an application wasn't investigated in the past, then we think it indeed shouldn't be investigated. This is very strict, since some applications maybe should have been scaled up but weren't. This option results in a dataset with only 2.2% positive labels, which leads to problems when evaluating the model. The capacity of HH suffices to investigate around 9% of all applications that are made, so they intend to investigate the top 9% applications suggested by the model. With only 2.2% positive labels, however, this means that even if the model is doing perfectly, it will always select at least 6.8% useless applications to get to 9%. This again means that the maximum achievable precision is less than 25% (2.2 over 9). One could argue that then only 2.2% of the applications should be sent to HH, but the business strongly expects the true percentage of 'onderzoekswaardigheid' to be higher. That expectation is supported by the baseline measurements of the working process, which show that a precision of around 50% is currently already attained. Hence, it's not possible to compare the model to the baseline with this label.

In option 2 we label screened rejections as positive, because if HH advises a rejection, that counts as a useful investigation. However, the rejections will also include some of the easy rejections described above. These kinds of applications should not be scaled up, but there is no way to distinguish between the 'easy' (should be sent to IC) and the 'hard' (should be sent to HH) rejections.

In option 3 we train our model with all screenings labeled negative, but evaluate it on a different label, where all screened rejections are labeled positive. The logic is that this way, it becomes more sensible

to compare the model to the baseline measurements, and the model looks promising. However, this label also cannot really be trusted for the reasons already laid out.

Therefore, we eventually decided to include only investigations and screenings by HH in the dataset and leave out the screenings by IC completely. This way the dataset contains only applications with a fairly reliable label.

HH screenings

This section describes the considerations around a label for the HH screenings. We eventually decided to exclude HH screenings from the dataset, so you may also skip this section. We did not trust the labels enough, since the working process that generated them was a substantial deviation from the normal process.

We have a different way to determine the label for screenings by HH. This is based on the concept of 'VPO' (Vervroegd Periodiek Onderzoek), where a HH screening receives a positive label if the application was put on VPO, and a negative label if it was not. Setting an application to VPO means granting the application and setting a future date at which the person should be re-investigated. This normally happens if something seems off, but it cannot be proven. During corona, this happened a lot, because it was impossible to go on house visits. Therefore, setting an application to VPO is actually really close to our concept of onderzoekswaardigheid.

6.2. Implementation

Handhaving advice

The source for the label is Sherlock. The advice given by HH after an investigation ('afboekcode') is found as a numerical code in the Proces table, column *srp_id*. The meaning of the codes is found in the table Sys_procesresultaat. The werkgroep helped us divide them into what should become a positive or negative label along the lines of reject/change/grant described earlier, or what should be filtered out completely, because we cannot tell. The latter is the case for instance when an application wasn't transferred properly from IC to HH, so they sent it back.

Linking investigations to applications

To make the codes into a label, the investigation must be linked to an application. Unfortunately, no direct link exists between applications/diensten and investigations. This, because it's never a specific dienst that's being investigated, but rather a person and the whole package of diensten they're receiving. This makes sense, because receiving one dienst can influence eligibility for another, so they cannot be investigated separately. Hence, we need to match investigations with applications using a rule of thumb. This rule of thumb was advised to us by Functioneel Beheer of Sherlock. In their experience, it's possible to match around 95% of investigations to an application in the following way.

First, the Proces table is filtered on investigations done at application time. This is indicated by the *spr_id* equal to 146 ('C Aanvraag'), as seen in the Sys_proces table. Second, the rule of thumb is applied: If a person was investigated within 14 days after their application was entered into the system, then the investigation was probably a result of that application. The entry date (*dtopvoer*) can lie after the application date (*dtaanvraag*), because some applications are still made on paper and entered into the system manually. It's used instead of the application date, because no one will start processing the application before it's been entered into the system.

There's a further complication to the rule of thumb, because it requires the original entry date of a dienst, since that is when the application was entered. However, the entry date in the Dienst table gets overwritten each time the dienst is updated, for example once a decision has been made. For this reason, the original entry date must be retrieved elsewhere. It can be found in the table Werkopdracht, which contains tasks for staff, including an automatically created task to process each new application. Its entry date is assumed to roughly match the original application entry date.

Determining how application was handled

IC screenings are identified by looking if a *pro_id* is linked to the application, because if there's none, it means no process belongs to it, which in turn means that HH was not involved.

HH screenings are identified by checking that an application *does* have a *pro_id*, and either:

- The process started between 23-03-2020 and 16-07-2020, the logic being that due to corona madness in this period all processes should be assumed screenings.

or

- The reason for the process was 'Aanvraag screening', the logic being that after 16-07-2020 this process reason was added with the purpose of tracking screenings by HH.

HH investigations are identified by checking that an application has a *pro_id*, but it doesn't match the criteria for the HH screenings.

Determining VPO

When setting an application to VPO, the employee can select at what date the re-investigation should take place. Whether or not an application was set to VPO can be determined by checking if the field with this date is filled. The relevant field is *pon_hercontroledatum* from Sherlock's Processtap_onderzoek table.

Result

For quite some applications that involved HH, around 2,000 out of 5,500, no label could be determined for various reasons. These applications are removed from the dataset. By far the largest chunk is because HH advised a 'hersteltermijn', which means to give the applicant another chance to submit additional information. In such cases, we can't tell whether it was good that the application was sent to HH. If the applicant submits the requested information and IC sends the application to HH once more, then the application will still show up in our dataset. If IC handles it themselves, then it won't be in our dataset.

The final dataset consists of some 3,400 applications. Roughly 55% have a positive label.

7. Model

7.1. Algorithm

The final algorithm is an Explainable Boosting Machine (EBM) without interactions to increase explainability. It's trained on 85% of the applications in the dataset and tested on the remaining 15%. The hyperparameter tuning and the model evaluation is done with a stratified k-fold cross-

validation grid search over 4 folds, using random seeds where necessary to make training runs reproducible.

The ROC AUC was used as the cross-validation metric (see next section for motivation). We checked the difference between train and validation set for each fold to identify overfitting. This was acceptable. We also checked the standard deviation of the ROC AUC between the folds to determine the stability of the model. The standard deviation was low (0.01), indicating that the model is stable.

The precise parameter grid with the values can be found in the repository, but the main hyperparameters that have been tuned are the following:

- learning_rate: Learning rate for boosting.
- max_leaves: Maximum leaf nodes used in boosting.
- max_bins: Maximum number of bins per feature for pre-processing stage.
- inner_bags: Number of inner bags.
- max_rounds: Number of rounds for boosting.
- min_samples_leaf: Minimum number of cases for tree splits used in boosting.

They were selected by researching their purpose online with a focus on performance and preventing overfitting.

The feature set is first decided using forward feature selection, then trimmed by leaving out any features that don't contribute to the first 95% of cumulative feature importance. This way, the model is greatly simplified - roughly one-fifth of the features make the cut. This is yet another way to reduce overfitting, besides also improving explainability and reducing the impact on citizens' privacy since less information is used.

We've also experimented with Gradient Boosting (XGB) and Random Forest (RF) as alternative algorithms. Our main interest was to find the optimal tradeoff between performance and explainability of the predictions. Our hypothesis was that XGB would perform best due to its increased complexity, whereas EBM would be the most explainable, with RF sitting somewhere in between. The explainability advantage of the EBM comes from the fact that the explanation of a prediction can be read directly from the model coefficients, whereas for the RF and XGB this must be approximated using a technique such as SHAP.

Table 1: Tradeoff between Gradient Boosting; Random Forest; and Explainable boosting machine.
Please be advised that the precision listed at the time when the model was selected (January 2022) and as the model has been reweighted since is no longer up-to-date.

	Explainable boosting machine (EBM)	Random forest (RF)	Gradient boosting (XGB)
Explainability	++	+	+
Comprehensibility	+	+-	-
Precision	61,5%	63,9%	59,2%

The conclusion of the experiments was that the RF performed best. This ruled out XGB immediately, because it's not more explainable than a RF. The choice was then between the more explainable but slightly less performant EBM and the less explainable but more performant RF. It's hard to understand or quantify precisely how much more trustworthy the explanations of the EBM are compared to the SHAP explanations of the RF. The business stakeholders then made the decision to go with the EBM, because they want to be as transparent as possible.

7.2. Metrics

Our main optimization criterion is the ROC AUC (Area under the ROC curve). Our data is a bit skewed (more negative than positive labels), and unlike other metrics, such as the accuracy, the AUC can be interpreted independently of the class distribution. Furthermore, the AUC is a good estimation of how well a model prioritizes applications for investigations: it can be interpreted as the probability that two randomly selected cases are ordered correctly (i.e., that an application which should be investigated has a higher score than an application which doesn't have to be investigated).

The AUC is calculated multiple times throughout the training and testing pipelines.

1. To estimate the quality of candidate parameter sets during the grid search.
2. To estimate the quality of the final model. This is done by calculating the AUC on the test data.

Our main evaluation criterion is the precision (also called 'hit rate' in the business-related documentation). This metric is the closest to what's measured in the business process, namely how many of the investigations by Handhaving had an impact on the outcome of the application, and more interpretable. This is interesting for two reasons: first, we want the capacity of Handhaving to be used as efficiently as possible, and second, investigations should be proportional: we don't want to bother citizens with an investigation if it's not necessary. The precision measures how many of the applications we select for Handhaving were indeed useful to investigate.

7.3. Evaluation

A portion (15%) of the data is kept aside as a test dataset. This is done in a stratified manner, so that the proportion of positive and negative labels in the train and test dataset are the same. Before splitting, the data is shuffled, so time order is not taken into account. This should not lead to data leakage (a.k.a. target leakage), because the features are created in such a way that only data that would be available at predict time is used, no data is used that links earlier and later applications by the same person, and no or only very few changes over time are expected.

The AUC on the test set is around 0.63, the precision on the test set is 64%. It's good to stress again that these figures are on a dataset that only contains HH investigations, no IC screenings. Since the HH investigations were in the past carried out after an IC forwarded the application due to irregularities, it's probable that the distribution of the features and the label is different between the train/test set versus the actual population of applications that the model will encounter in practice. A pilot is therefore essential to evaluate the model more.

The detailed statistics of the model can be found in the model statistics file.

8. Implementation

8.1. Way of working

It’s technically enforced that merge requests can be merged only when one person other than the author has reviewed and approved them. Also, it’s not possible to push to the development or master branch directly.

Best practices have been followed to create cleaner, more readable and efficient code, focusing on reusability, simplicity and maintainability. Internal packages were created to structure the project, perform the bias analysis and handle the Azure interaction. This isolates the project-related code from the structural code.

The following pre-commit hooks have been used to ensure code quality:

- Isort: sort imports alphabetically, and automatically separated into sections and by type.
- Black: code formatter, makes code review faster by producing the smallest diffs possible.
- Flake8: toolkit for checking your code base against coding style (PEP8), programming errors (like “library imported but unused” and “Undefined name”) and to check cyclomatic complexity.
- Mypy: static type checker for Python that aims to combine the benefits of dynamic (or "duck") typing and static typing.

The code is also checked by unit tests on the most complex and crucial parts, and the API has a healthcheck endpoint that is called periodically. This endpoint checks the availability of the data, of the endpoint itself, and the operationality of the model. These tests are performed automatically at every deployment, and in case of test failure the deployment is interrupted.

The API furthermore contains data quality checks on the data received in the score endpoint, and on the response.

When training the model, a markdown file with a model summary (some input data statistics, performance metrics, feature importances, et cetera) gets generated. This file is saved as an artifact with all registered models on Azure.

8.2. Deployment

The model is deployed on Azure. The input and output pipelines are maintained by Datateam Sociaal. For more information, please refer to the “architectuurnotitie”.

The model output is a score between 0 and 1 for each application. The API response consists of:

- The score
- EBM native explanation of each individual prediction
- Feature values
- Version number of the model (referring to the model name and version as in AzureML)
- Subject number of the applicant

The output is shown to the end users in the Dataverkenner.

On an infrastructure level, the following things get logged (standard based on the Azure building blocks of the CCC):

- Which operations were done on which resources
- Who initiated them

- When this took place (date and time)
- What the result of the operation was

On the model/API level, we log:

- API requests including date, time, source, response code, (request JSON, response JSON)
- Deployments including the new model version
- Start/completion time of various events in the scorer and preprocessor
- Warning in case of missing input data

9. Further documentation

The code of the model can be found in the project's Git repository on Azure. This also includes the code change history, a verbal changelog, and the project's requirements in terms of software/packages.

The bias analysis carried out on the data and the model is described in the bias analysis documentation.

Information on the technical architecture of the model's development and deployment can be found in the architectuurnotitie.

The beheerplan documents how the model will be monitored and maintained.