Energy effectiveness of a mortgage portfolio

Jakub Vondráček, Daniel Till

27.12.2022

Contents

1 Introduction

Due to the changing climate and looming energy crisis, the energy efficiency of buildings is more important than it ever was before. Particularly, it might be a deciding factor when applying for mortgages in the UK in the future due to regulations, since according to Open Data Communities, close to 40% of the UK's energy consumption and emissions are produced from the heating and usage of buildings.

To manage risk well, banks need to be able to predict the energy rating for buildings for which the energy rating has not yet been determined.

2 Assignment

Our task is to build a model that predicts the energy rating of a building based on available data. Particularly, we were tasked to compare several models that can be used to predict the energy rating and then choose the one with the best accuracy.

Furthermore, the data that are available to the client are rather limited, containing only a few regressors. We were thus also tasked to identify potentially useful regressors that are not currently used by the client and that could be included in the model to increase the accuracy.

3 Data analysis

We obtained a dataset from Open Data Communities on energy performance of buildings located in England and Wales. The dataset contains the energy rating certificates for over 20 million buildings in 341 regions. In total, the dataset is over 30GB in size, which does not fit into RAM in most personal computers[1].

The variables located in the raw dataset and their description can be found in Open Data Communities - Variable Glossary.

The most important thing to note here is that the dataset consists of many categorical variables and only a modest number of numerical variables.

3.1 Data cleaning

Due to the gigantic nature of the dataset, we had to resort to partitioning when preprocessing the data, which added significant complexity to the preprocessing process.

The most demanding preprocessing step of the data was parsing categorical variables correctly. The dataset consists of many fields with some sort of text input, which describes the given variable. Some of these features are quite well behaved and didn't need much manipulation, but some of them consisted of hundreds and thousands distinct values, which we needed to parse and group to obtain a reasonable training dataset.

While we intentionally keep this section on the data cleaning process short as it is not particularly interesting from the modelling perspective, we must note that it was probably the most time demanding part of the whole project.

3.2 Data exploration

In this section, we pose our findings for several (mostly categorical) variables found in the dataset.

The most interesting variable is `CURRENT_ENERGY_RATING`, which is the dependent variable which we want to predict. We will refer to it as *current energy rating* or *target* interchangeably. Particularly, we must notice that the target is an ordinal variable – it represents 7 energy rating classes, A, B, C, D, E, F, G, where A is the best and G is the worst.

[1] which was a significant limiting factor

Value	Count	Frequency
D	8650444	43.8%
C	5789245	29.3%
E	3602636	18.3%
F	919699	4.7%
B	476051	2.4%
G	281359	1.4%
A	11011	0.1%

Table 1: `CURRENT_ENERGY_RATING` – frequencies in the cleaned dataset.

Table 1 shows the counts and frequencies of the target classes in the cleaned dataset. We must note that the classes are significantly imbalanced. We are thus tasked with an imbalanced ordinal classification problem.

We were tasked to create a suitable model by using only the regressors that the client is able to obtain when faced with predicting the energy rating for a new building. These regressors are `PROPERTY_TYPE` (see Table 2), `BUILT_YEAR`[2] (which is also referred to later as `BUILDING_AGE_CLASS`), `ADDRESS`, `POSTCODE`, `FLOOR_AREA` and `NUMBER_HABITABLE_ROOMS`, see Sections 7.2 and 7.3 for their description.

Value	Count	Frequency
House	12218338	61.9%
Flat	5128837	26.0%
Bungalow	1854808	9.4%
Maisonette	519373	2.6%
Park Home	9089	0.1%

Table 2: `PROPERTY_TYPE` – frequencies in the cleaned dataset.

We decided to consider number of habitable rooms as a numerical variable, since while the values are discrete, by considering it as a categorical variable, we would lose the inherent ordinality[3], see Table 3.

[2] which is a numerical variable that represents the year the building was built, parsed from the `CONSTRUCTION_AGE_BAND` variable

[3] we experimented with inclusion of this and similar ordinal variables (`BUILT_YEAR`) as either categorical or numerical on a subset of the dataset and inclusion of the ordinal variables as numerical led to slightly better performance. Note that while the variables `BUILT_YEAR` and `NUM_HABITABLE_ROOMS` are ordinal, the differences between each level are meaningful, i.e. we know that a building with 3 rooms has one more room compared to a 2 room building, in contrast to the current energy rating, where that levels have no such interpretation. Furthermore, including them as numerical variables means that we did not have to use one-hot encoding for these variables, which reduced the size of the dataset in memory as we were running into memory problems

Value	Count	Frequency
5	4528211	23.0%
4	4471494	22.7%
3	4163255	21.1%
2	2571045	13.0%
6	1775646	9.0%
7	959924	4.9%
8	476802	2.4%
1	382361	1.9%
9	212718	1.1%
10	96686	0.5%
Other	92303	0.5%

Table 3: `NUM_HABITABLE_ROOMS` – frequencies in the cleaned dataset.

`ADDRESS` was dropped from the dataset since we couldn't find a way to include it in a sensible way, that would not be already included in `POSTCODE`. `POSTCODE` was used to augment the dataset with several other variables which substitute it in some way, see the next section. Unfortunately, we couldn't use `POSTCODE` itself in the model, since due to the large number of distinct Post codes in the UK (over 1.8 million), we were running into memory problems with some of the considered models due to the need for creating dummy variables (we refer to this as one-hotting), which yielded an insanely large dataset with the inclusion of `POSTCODE`[4] as a categorical variable.

`BUILT_YEAR` is a numerical variable which consists of discrete classes (since the data is not precise) – again similarly as with the number of habitable rooms, we include it as a numerical variable to not lose the inherent ordinality. For the frequencies of `BUILT_YEAR`, see Table 4.

Value	Count	Frequency
1965	3527999	17.9%
1925	3165898	16.0%
1945	2765658	14.0%
1975	2468852	12.5%
1900	2319324	11.8%
1990	1395788	7.1%
1980	1269953	6.4%
2000	1033550	5.2%
2005	867870	4.4%
1995	798667	4.0%

Table 4: `BUILT_YEAR` – frequencies in the cleaned dataset.

[4]we also ran into problems when splitting the dataset into train, validation and test sets, as since there are so many Post codes, it was impossible to obtain a split such that at least one building from each Post code was included in the train dataset, which would prevent us from predicting on the test dataset, as the model would get a level of Post code that was not seen in the training data

3.3 Augmenting the dataset

The client posed a question that it might be interesting to group buildings together in clusters based on their location, as it might be a significant predictor of the energy rating (usually, buildings that are located close to each other are built in the same manner and sometimes also at the same time, meaning that they might suffer from the same problems and thus knowing the energy rating of a building that is close to the building which we want to predict the rating for might be a significant predictor).

We proposed a way to calculate GPS coordinates from the address using the geopy package, which unfortunately provided us only with an open source API for fetching the GPS coordinates from OpenStreetMap, which was extremely slow (taking about 1s per building). This obviously doesn't scale to the size of our dataset. Due to the fact that we didn't have access to a proper API for fetching the GPS coordinates, we abandoned this idea.

Fortunately, when trying to calculate the GPS coordinates and the distance between buildings based on the GPS coordinates, we noticed a peculiar thing. For the buildings located in the same Post code, the distances between them were very small. We thus found out that the Post code is the lowest level of granularity of regional partitioning in the UK, always containing only a few buildings that are very close to each other.

We thus decided to use Post code as a proxy for the distance between buildings and for each building in the dataset, we calculated the current energy rating proportions of buildings located in the same Post code (excluding the current property). We included these variables in the dataset as

$$\texttt{A\_prop\_POSTCODE}, \ldots, \texttt{G\_prop\_POSTCODE}.$$

This also took a significant amount of time for such a large dataset, but we were able to compute these proportions within days of runtime on a personal computer. Of course, it could be argued, that for a building that is on the very edge of the Post code, buildings in other Post codes that are close might also be relevant for the energy rating of the currently considered building, but we do not consider such effects.

We further computed the variables `POSTCODE_PROPORTIONS_ARE_RELIABLE_IND`, `IS_EPC_LABEL_BEFORE_2008_INCL` and `POSTCODE_COUNT` for each building in the dataset, see Sections 7.2 and 7.3 for their description.

All of the aforementioned variables combined should exhaustively substitute not using the `POSTCODE` variable explicitly and should contain more information that just the categorical level of `POSTCODE` itself.

3.4 Small and Big dataset

As we mentioned before, we need to train a model that is useful for the client given the limited data they have available and also potentially discover which new regressors may be useful for the prediction of the energy rating.

For the task of training a model for prediction using the regressors specified by the client, we created a dataset termed *small*, that consists only of the variables

- BUILDING_AGE_CLASS (this is just another name for BUILT_YEAR),
- FLOOR_AREA,
- NUMBER_HABITABLE_ROOMS,
- PROPERTY_TYPE,
- BUILT_FORM,
- A_prop_POSTCODE, ..., G_prop_POSTCODE,
- POSTCODE_PROPORTIONS_ARE_RELIABLE_IND,
- IS_EPC_LABEL_BEFORE_2008_INCL,
- POSTCODE_COUNT,
- LOCAL_AUTHORITY_LABEL,

where apart from the explanatory variables explicitly requested by the client we added the LOCAL_AUTHORITY_LABEL as it is always available when ADDRESS is available (and we weren't able to include ADDRESS sensibly), BUILT_FORM as it together with the PROPERTY_TYPE gives a structured description of the property[5] and further the variables POSTCODE_COUNT, IS_EPC_LABEL_BEFORE_2008_INCL and POSTCODE_PROPORTIONS_ARE_RELIABLE_IND, see Sections 7.2 and 7.3 for their description.

For the purposes of finding which new regressors may be useful to improve the model, we created a dataset which contained all the available features, which we term *big*.

Remark. *Obviously, the big dataset may contain regressors that are used directly to calculate the energy rating. We thus iteratively dropped regressors that were very valuable (in terms of feature importance, more on that in Section 5.4), until the models did not provide perfect predictions from the big dataset, but rather until we got a dataset that provided predictions that were reasonably better compared to the small dataset, but not unachieveably precise. We do not explicitly enumerate the variables used in the big dataset here, these can be found in the feature importance figures in Section 5.4.*

3.5 Imbalanced or balanced?

When training the models we were faced with a dilemma, whether to train the model on the imbalanced dataset or introduce class weights to adjust the model to not only predict well the majority class but to also focus on the minority classes. To write this formally, if we have a single observation $(\mathbf{x}^T, y)^T$ that belongs in class c with class weight w_c, then

[5]it consists only of 6 levels and it is available for over 98% of buildings in the dataset (we classified the rest as UNKNOWN) and should be very easily and cheaply obtainable by just asking the owner of the property and in the worst case when it is not available, the UNKNOWN level can be used, see Section 7.2

the loss obtained from this particular observation is $w_c l(\mathbf{x}, y)$, where $l(\cdot)$ is a given loss function. This adjustment is then performed for all observation in the dataset. Using these class weights, we can make the model focus more on minimising the loss over the minority classes by giving them a higher weight.

In Section 5, we give the results of training the models for both the imbalanced dataset and then training models where we adjust class weights in such way that the models could be considered to be trained on a dataset where classes are balanced.

When presenting our intermediate findings from both approaches to the client, we discovered that the most important metric they care about is accuracy (predicting the energy rating for as many buildings correctly as possible) and that they are working with a data distribution that is a little different compared to the distribution in our training dataset. Particularly, we were given the distribution of target classes that the client expects to receive (when faced with prediction) and asked to train the best performing model with adjustment for this distribution. We adjusted the class weights based on this predefined distribution and the distribution in the training dataset so that the model gives more weight to the classes that are underrepresented in our distribution compared to the predefined distribution and less weight to the classes that are overrepresented in our distribution compared to the predefined distribution (it can be thought of as training a model on a dataset where the proportions of each class in the target variable are given by the predefined distribution). The predefined distribution is given in Table 5 (compare with Table 1) and the results of training the models with the predefined distribution are also given in Section 5.

Value	Frequency
D	38.9%
C	22.7%
E	17.2%
F	3.9%
B	16.3%
G	0.7%
A	0.3%

Table 5: Distribution of target predefined by the client.

3.6 Preparing the data for training

In order to be able to evaluate the proposed models correctly, we needed to split the dataset into three disjoint sets, a training, validation and test sets. This was done using stratified[6] random sampling, where due to the size of the whole dataset, we were able to only utilize 25%[7] of the whole dataset as the training set. The validation set also consisted of 25% of the total observations and the remaining 50% of observations were

[6]which assures that the proportions of each target class are equal in each set

[7]this is a significant limitation, as being able to train the model on a larger dataset might lead to much better results

used as the test set. The validation set was used for hyperparameter optimization for gradient boosting models and support vector machine models and the test set was then used to evaluate the models.

We note here that splitting the dataset into train, validation and test sets took a lot of trial and error due to the large amount of categorical variables with many levels in the dataset, since we needed each level of the categorical features in the training dataset to be able to predict on the test set.

4 Models

We have chosen to try and compare three different models that can be used to solve the task at hand, namely the ordinal regression model, support vector machines and gradient boosted decision trees.

Let $(\mathbf{x}_i^T, y_i)^T \overset{i.i.d.}{\sim} (\mathbf{x}^T, y)^T$, $i = 1, ..., N$, where N is the size of the dataset. In the context of our problem, y is the target variable and $\mathbf{x} \in \mathbb{R}^k$ are the explanatory variables. Instead of the energy rating classes $A, \ldots, G$ we will use general classes $0, \ldots, R$ with ordinal relationship $0 < \cdots < R$, i.e. $R = 6$ where applicable for simplicity of notation.

4.1 Ordinal Regression Model

This section is based on [4].

The ordinal regression model is the only model in our repertoire that actually takes into account the inherent ordinality present in the target variable. It considers an unobservable latent variable y^*, which is linked to the explanatory variables (regressors) by a linear model

$$y_i^* = \mathbf{x}_i^T \boldsymbol{\beta} + \varepsilon_i, i = 1, ..., N, \tag{1}$$

where $\varepsilon_i \overset{i.i.d.}{\sim} \varepsilon$ are i.i.d. random variables with zero mean and a distribution function F and $\boldsymbol{\beta} \in \mathbb{R}^k$ is a vector of coefficients. Observable dependent variable y is multi-categorical with the values:

$$y_i = \begin{cases} 0 & \text{for} \quad y_i^* \le m_1, \\ 1 & \text{for} \quad m_1 < y_i^* \le m_2, \\ 2 & \text{for} \quad m_2 < y_i^* \le m_3, \\ \vdots & \\ R & \text{for} \quad m_R < y_i^*, \end{cases}$$

$i = 1, ..., N$ and where the thresholds $m_1, \ldots, m_R$ are in addition to $\beta_1, \ldots, \beta_k$ unknown model parameters. We can then write:

$$\begin{aligned} &P(y_i = 0|\mathbf{x}_i, \boldsymbol{\beta}, \mathbf{m}) = F(m_1 - \mathbf{x}_i^T\boldsymbol{\beta}), \\ &P(y_i = 1|\mathbf{x}_i, \boldsymbol{\beta}, \mathbf{m}) = F(m_2 - \mathbf{x}_i^T\boldsymbol{\beta}) - F(m_1 - \mathbf{x}_i^T\boldsymbol{\beta}), \\ &P(y_i = 2|\mathbf{x}_i, \boldsymbol{\beta}, \mathbf{m}) = F(m_3 - \mathbf{x}_i^T\boldsymbol{\beta}) - F(m_2 - \mathbf{x}_i^T\boldsymbol{\beta}), \\ &\vdots \\ &P(y_i = R|\mathbf{x}_i, \boldsymbol{\beta}, \mathbf{m}) = 1 - F(m_R - \mathbf{x}_i^T\boldsymbol{\beta}), \end{aligned}$$

where F is the distribution function of the residual component ε of the model specified in Equation 1.

Estimation of the model is done by the maximum likelihood method. The log likelihood function has for our case (logit model) the form:

$$
\begin{aligned}
L(\boldsymbol{\beta}, \mathbf{m}) &= \sum_{i=1}^{N} \sum_{r=0}^{R} 1_{[y_i=r]} \cdot \log\left(\mathbb{P}(y_i = r|\mathbf{x}_i, \boldsymbol{\beta}, \mathbf{m})\right) \\
&= \sum_{i=1}^{N} \sum_{r=0}^{R} 1_{[y_i=r]} \cdot \log\left(\frac{e^{m_{r+1}-\mathbf{x}_i^T\boldsymbol{\beta}}}{1+e^{m_{r+1}-\mathbf{x}_i^T\boldsymbol{\beta}}} - \frac{e^{m_r-\mathbf{x}_i^T\boldsymbol{\beta}}}{1+e^{m_r-\mathbf{x}_i^T\boldsymbol{\beta}}}\right)
\end{aligned}
$$

where $1_{[y_i=r]}$ is the indicator that observation i belongs to class r and where we formally put $m_0 = -\infty$ and $m_{R+1} = +\infty$.

After the model is estimated, we can use the estimated $\hat{\mathbf{m}}$ and $\hat{\boldsymbol{\beta}}$ to predict class probabilities for a new observation $\mathbf{x}_{new}$ as

$$
\begin{aligned}
P(y_{new} = r|\mathbf{x}_{new}, \hat{\boldsymbol{\beta}}, \hat{\mathbf{m}}) &= F(\hat{m}_{r+1} - \mathbf{x}_{new}^T\hat{\boldsymbol{\beta}}) - F(\hat{m}_r - \mathbf{x}_{new}^T\hat{\boldsymbol{\beta}}) \\
&= \frac{e^{\hat{m}_{r+1}-\mathbf{x}_{new}^T\hat{\boldsymbol{\beta}}}}{1+e^{\hat{m}_{r+1}-\mathbf{x}_{new}^T\hat{\boldsymbol{\beta}}}} - \frac{e^{\hat{m}_r-\mathbf{x}_{new}^T\hat{\boldsymbol{\beta}}}}{1+e^{\hat{m}_r-\mathbf{x}_{new}^T\hat{\boldsymbol{\beta}}}}, \quad r = 0, \ldots, R.
\end{aligned}
$$

Predictions of a particular class are then obtained by taking the argmax of the obtained class probabilities.

The model we used for the small dataset can then be written in the ordinal regression context as

$$
\begin{aligned}
\text{CURRENT\_ENERGY\_RATING} \sim\, &\text{PROPERTY\_TYPE} + \text{BUILT\_FORM} \\
&+ \text{FLOOR\_AREA} + \text{NUMBER\_HABITABLE\_ROOMS} \\
&+ \text{A\_PROP\_POSTCODE} + \ldots + \text{G\_PROP\_POSTCODE.} \\
&+ \text{POSTCODE\_PROPORTIONS\_ARE\_RELIABLE\_IND} \\
&+ \text{IS\_EPC\_LABEL\_BEFORE\_2008\_INCL} \\
&+ \text{POSTCODE\_COUNT} + \text{BUILT\_YEAR} \\
&+ \text{LOCAL\_AUTHORITY\_LABEL}
\end{aligned}
$$

and was estimated using the *R* library *MASS*.

4.2 Support vector machine

Support vector machine (SVM) is a supervised machine learning algorithm useful for binary classification, which learns an optimal boundary between data points belonging to each class. This section is based on [7].

Strictly speaking, in the SVM, the objective is to find such a boundary that divides the data points and maximises the distance between the boundary and the data point from each class[8] that is the closest to the boundary – there exist always 2 such points, one for each class. These points are called support vectors.

Support vector machine doesn't support multiclass classification natively, just binary classification, so a trick is used for the multiclass case. For each class, a binary SVM is constructed that finds the optimal boundary between the given class and the rest of

[8]remember we talk about binary classification, so there are only 2 classes considered

the data – this can be thought of as a one-vs-rest approach. Next we only focus on the binary classification (one-vs-rest) case and we come back to handling the multiclass classification at the end of this section.

We note here that the SVM problem formulation can be written in such a way that it can be solved explicitly using a chosen mathematical programming solver. Unfortunately, due to the large amount of data and our limited computing capacity, we had to resort to a stochastic gradient descent approach using the *sklearn* library in Python. Let us give a short introduction to how the algorithm works when using stochastic gradient descent for the binary classification case.

In general, a discriminant function

$$f(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}),$$

is used where $\Phi : \mathbb{R}^k \to \mathbb{R}^j$, $\mathbf{w} \in \mathbb{R}^j$ and a variable y_i' is considered, such that $y_i' = 1$ if $y_i = r$ and $y_i' = -1$ otherwise. In our case, we use the linear discriminant function

$$f(\mathbf{x}) = \mathbf{w}^T (1, \mathbf{x}^T)^T = \mathbf{w}^T_{[1:]}\mathbf{x} + w_1,$$

where $w_1 \in \mathbb{R}$ is the first term in the vector $\mathbf{w}^T$ and $\mathbf{w}^T_{[1:]}$ is the vector $\mathbf{w}^T$ without the term w_1.

The problem of finding the optimal boundary can then be formulated as minimization[9] of the risk function $R(\mathbf{w})$

$$\min_{\mathbf{w}} R(\mathbf{w}) = \min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} l(y_i', f(\mathbf{x}_i)) + \lambda J(\mathbf{w}),$$

where l is a loss function, J is a penalty function and $\lambda > 0$ is a regularization parameter.

In our case, to be able to obtain also probability predictions[10], we have chosen the modified Huber loss as the loss function, see [7, p. 1] and [12, p.74][11]. The modified Huber loss is then given as

$$l(y_i', f(\mathbf{x_i})) = \begin{cases} -4y_i' f(\mathbf{x}_i), & y_i' f(\mathbf{x}_i) < -1, \\ (1 - y_i' f(\mathbf{x}_i))^2, & y_i' f(\mathbf{x}_i) \in [-1, 1), \\ 0, & y_i' f(\mathbf{x}_i) \geq 1, \end{cases}$$

where we add the subscript i so that the values can be interpreted in the context of the minimisation of the risk function above. The minimisation is then performed using stochastic gradient descent.

We described the construction of a binary SVM, the multiclass case consists of building $R+1$ one-vs-rest classifiers, one for each class. Let us now describe how the probability predictions are computed in the *sklearn* Python library, where the implementation is

[9] see also Scikit-learn documentation

[10] in the *sklearn* library, the probability predictions are implemented only for the modified Huber loss with the stochastic gradient descent implementation

[11] more information can be found here

based on the paper [11]. For each of these binary classifiers, the probability predictions $p_{r,new}$ for a new observation $\mathbf{x}_{new}$ are given as

$$p_{r,new}(\mathbf{x}_{new}) = \frac{\max(\min(\mathrm{dist}_r(\mathbf{x}_{new}), 1), -1) + 1}{2}, r = 0, ..., R, \tag{2}$$

where $\mathrm{dist}_r(\mathbf{x}_{new})$ is the normalised[12] signed distance to the boundary in the binary classifier for class r-vs-rest. The obtained probabilities $p_{r,new}, r = 0, ..., R$ are then normalised so that they sum up to 1 to obtain a proper probability distribution, see the following Equation:

$$P(y_{new} = r|\mathbf{x}_{new}) = \frac{p_{r,new}(\mathbf{x}_{new})}{\sum_{j=0}^{R} p_{j,new}(\mathbf{x}_{new})}, r = 0, ..., R.$$

Predictions of a particular class are then obtained by taking the argmax of the obtained class probabilities $P(y_{new} = r|\mathbf{x}_{new}), r = 0, ..., R$.

The above approach can be interpreted as using the distance to the boundary in each classifier as a proxy for how probable it is that the point belongs to the particular class. This of course needs proper justification, we refer the interested reader to [11] and [13, Appendix B] and note that this is the approach used in the *sklearn* implementation, which is one of the most widely used machine learning libraries, see [10].

For training the SVM model, the hyperparameter λ was chosen using hyperparameter optimization and the penalty function is $J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$, where $\|\cdot\|_2$ denotes the L^2 norm.

[12]divided by the absolute value of the maximum distance from the boundary on the side corresponding to the current observation $\mathbf{x}_{new}$ of the boundary encountered during training. Note that the obtained normalised signed distance can be larger than 1 if the new observation $\mathbf{x}_{new}$ is further from the boundary than the furthest point from the boundary on that side encountered during training, which is why the clipping to [-1,1] is performed in Equation 2.

4.3 Gradient boosting

This section is based on [5, Section 4.6.] and [3] and [6, Chapter 10].

Gradient boosting is a machine learning model based on decision trees. It consists of an ensemble of weak learners (decision trees), which come together into a single strong learner. The training procedure consists of M steps, i.e. M trees to be built. In each step, a decision tree is constructed that greedily improves the prediction from the previous step.

The goal is to obtain an estimate of a function $f(\mathbf{x}) : \mathbb{R}^k \to \mathbb{R}^{R+1}$,

$$f(\mathbf{x}) = (f_0(\mathbf{x}), ..., f_R(\mathbf{x})),$$

that minimizes a loss function $L(y, f(\mathbf{x}))$[13]. Using the notation developed earlier, for the target and explanatory variables $(\mathbf{x}_i^T, y_i)^T, i = 1, ..., N$, the loss function[14] can be written as

$$L(\{\{y_i^{(j)}, f_j(\mathbf{x}_i)\}_{j=0}^R\}_{i=1}^N) = -\left[\sum_{i=1}^N \sum_{j=0}^R y_i^{(j)} \log p_j(\mathbf{x}_i)\right] + \Omega(\{f_j)\}_{j=0}^R), \tag{3}$$

where $y_i^{(j)}$ is an indicator that y_i is equal to j, i.e. $y_i^{(j)} = 1_{[y_i=j]}$,
$p_j(\mathbf{x}_i) = P(y_i^{(j)} = 1|\mathbf{x}_i)$, $j = 0, \ldots, R$, $\{a_j\}_{j=0}^R = (a_0, ..., a_R)$, Ω is a regularisation term that penalises the complexity of $\{f_j\}_{j=0}^R$ and $p_j(\mathbf{x}_i)$ are calculated using the softmax function as

$$p_j(\mathbf{x}_i) = \frac{\exp(f_j(\mathbf{x}_i))}{\sum_{z=0}^R \exp(f_z(\mathbf{x}_i))}. \tag{4}$$

Each function f_j represents the output of a decision tree and is of the form $f_j(\mathbf{x}) = w_{j,q_j(\mathbf{x})}$, where $q_j : \mathbb{R}^k \to \{1, ..., T_j\}$, where T_j is the number of leaves in the tree j. The decision tree maps (the mapping can be highly non-linear) $\mathbf{x}$ to one of the leaves, where it is assigned a weight (the weights are among the learned parameters[15]). We denote the weight $w_{j,z}, z \in \{1, ..., T_j\}, w_{j,z} \in \mathbb{R}$ is the weight at leaf z in tree j.

We have chosen to use the LightGBM implementation [8, version 3.3.3.]. In this implementation, the regularisation term $\Omega(\{f_j)\}_{j=0}^R)$ in Equation 3 can be written in the form

$$\Omega(\{f_j)\}_{j=0}^R) = \sum_{j=0}^R \left[\lambda_1 \sum_{z=1}^{T_j} |w_{j,z}| + \frac{1}{2}\lambda_2 \sum_{z=1}^{T_j} w_{j,z}^2\right],$$

where T_j is the number of leaves of the tree f_j, $w_{j,z}$ are the weights at the leaves of the tree and λ_1 and λ_2 are hyperparameters, which control the strength of the regularisation.

[13] note that here we use the definion of L as a function of a general $(\mathbf{x}^T, y)^T$, while in Equation 3 we consider it is the loss calculated over the whole dataset

[14] see also Scikit-learn documentation and XGBoost documentation

[15] along with the structure of the tree, number of leaves etc.

The regularisation used here is a combination of L^1 and L^2 regularisation, which is sometimes referred to as ElasticNet.

Let $f_k^{(0)}(\mathbf{x}) = 0$, $k = 0, \dots, R$, where $f_k^{(0)}(\mathbf{x})$ is an estimate of $f_k(\mathbf{x})$ at step 0. Analogously $f_k^{(m)}(\mathbf{x})$ is an estimate of $f_k(\mathbf{x})$ at step $m = 1, \dots, M$.

The update with a new tree can be summarised as

$$f_k^{(m)}(\mathbf{x}) = f_k^{(m-1)}(\mathbf{x}) + \alpha f_k^{(m),new}(\mathbf{x}), k = 0, ..., R,$$

where $f_k^{(m),new}$ is the new tree that is obtained as a minimiser[16] of the loss function $L(\{\{y_i^{(j)}, f_j^{(m-1)}(\mathbf{x}_i) + f_j^{(m,new)}(\mathbf{x}_i)\}_{j=0}^R\}_{i=1}^N)$, where $f_j^{(m-1)}, j = 0, ..., R$ is known (the tree from the previous step) and α is a hyperparameter (learning rate).

Predictions of probabilities from the whole ensemble $\left\{p_k^{(M)}(\mathbf{x})\right\}_{k=0}^R$ could be computed from the final estimates $\left\{f_k^{(M)}(\mathbf{x})\right\}_{k=0}^R$ using (4), so

$$P(y = k|\mathbf{x}) = p_k^{(M)}(\mathbf{x}) = \frac{\exp(f_k^{(M)}(\mathbf{x}))}{\sum_{j=0}^R \exp(f_j^{(M)}(\mathbf{x}))}, \quad k = 0, \dots, R.$$

Predictions of a particular class are then obtained by taking the argmax of the obtained class probabilities. The hyperparameters were found using hyperparameter optimization.

4.4 Model evaluation

In this section, we describe model evaluation. By a confusion matrix we will understand a matrix $\{c_{ij}\}_{i,j \in \{0,\dots,R\}}$, where c_{ij} is the number of samples that were classified as class j when the true class was i. We will use following terms for every class $i \in \{0, \dots, R\}$:

- **True Positives** (TP_i): c_{ii},
- **True Negatives** (TN_i): $\sum_{j,k \in \{0,\dots,R\} \setminus \{i\}} c_{jk}$,
- **False Positives** (FP_i): $\sum_{j \in \{0,\dots,R\} \setminus \{i\}} c_{ji}$,
- **False Negatives** (FN_i): $\sum_{j \in \{0,\dots,R\} \setminus \{i\}} c_{ij}$.

[16]we intentionally keep the minimisation step vague so that we do not have to develop needlessly complex notation, as the procedure requires specification of how exactly the trees are constructed, which gets very involved. This is out of the scope of this text. The interested reader can find details in [3, Section 2.2.] and in [5, Section 4.6.] and [6, Chapter 10]

Then we define the following metrics for every class $i \in \{0, \ldots, R\}$:

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i},$$

$$\text{Recall}_i = \frac{TP_i}{TP_i + FN_i},$$

$$\text{F1 Score}_i = 2\frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i},$$

where F1 Score is the harmonic mean of Precision and Recall and for each of these metrics, a higher value is preferred.

For an observation $(\mathbf{x}^T, y)^T$ we define the Ranked Probability Score as given in [9, p. 1]

$$\text{RPS}((\mathbf{x}^T, y)^T) = \frac{1}{R+1}\sum_{k=0}^{R}\left[\left(\sum_{j=0}^{k} p_j(\mathbf{x})\right) - \left(\sum_{j=0}^{k} y^{(j)}\right)\right]^2,$$

where $p_j(\mathbf{x})$ is the predicted probability that the observation belongs in class j and $y^{(j)} = 1_{[y=j]}$. Then for a given class $i \in \{0, ..., R\}$, the Ranked Probability Score obtained over class i can be calculated as follows:

$$\text{RPS}_i = \frac{1}{|i|}\sum_{((\mathbf{x}_l^T, y_l)^T)\in i} \text{RPS}((\mathbf{x}_l^T, y_l)^T),$$

where $|i|$ denotes the number of observations in class i and the subscript $((\mathbf{x}_l^T, y_l)^T) \in i$ denotes that the sum is taken over observations that belong in class i. RPS_i is the mean RPS over observations that belong in class i. The Ranked Probability Score is a metric that captures the inherent ordinality of the data. Particularly, the values of RPS are always in $[0, 1]$ and a lower value is better. Further we define

$$\text{Accuracy} = \frac{\sum_{i\in\{0,\ldots,R\}} c_{ii}}{\sum_{i,j\in\{0,\ldots,R\}} c_{ij}},$$

which is the proportion of observations that were classified in the correct class out of all observations. Obviously, the accuracy is also always in [0,1] and a higher value is better.

When presenting our intermediate results to the client, we learned that they are mainly interested in the accuracy of the model, i.e. being able to predict the target for as many buildings correctly as possible while using the distribution of data in the dataset. We further use the Ranked Probability Score to evaluate the models to capture the inherent ordinality of the target variable.

4.5 Computational difficulties

The only model which we had computational difficulties with using the small dataset is the ordinal regression model. It seems that training the ordinal regression model is very memory intensive. This model was the only one implemented in R and calculated on the cluster Sněhurka.

Particularly, we considered two versions of each model - one trained on the dataset without any adjustments and one trained with additional class weights to treat the imbalancedness. Unfortunately, we were able to train only the unbalanced version of the ordinal regression model due to computational problems. We were able to train this model in a queue which allowed 6 hours of computation.

To train the ordinal regression with balanced classes, we tried our best to obtain a working implementation, but it seems that this kind of problem is not encountered much in practice, as we couldn't find a reasonable implementation that allowed for setting the class weights explicitly. Particularly, the ordinal regression packages we tried (*MASS* and *orm* in *R*) did not allow for setting the class weights explicitly and we had to resort to oversampling[17] as our only option. Unfortunately, this led to an exceedingly large size of the dataset, which was still possible to be contained in memory, but both the packages listed above failed when training. Particularly, they failed in the 6 hour computation queue due to insufficient time limit. Calculation in any queue which allowed for more than 6 hour of computation failed due to R error associated with the inability of allocation of a too large vector. This error means we want to use more memory than R can use. Unfortunately, we were unable to fix this issue. We thus only consider the unbalanced version of the ordinal regression model. Remaining models were trained without any problems[18] on the small dataset.

4.6 Data manipulation

Particularly for the SVM model, we had to perform nontrivial data manipulation in order to obtain reasonable predictions. Such manipulation consisted of one hot encoding categorical variables and standardising[19] numerical variables..

The other two models did not suffer from these problems and we were able to train the models on the small dataset without any data manipulation.

[17]undersampling would not let us predict on the whole test dataset, since the model would not see some levels of the categorical variables

[18]the ease of use must be also considered when considering that the model may be deployed in production and a nontrivial effort will be expended on supporting the model. Particularly, a model that is not problematic and easy to train and use is cheaper to maintain due to less time spent supervising the model and might thus be preferred to something much more complex and difficult to work with

[19]subtracting the sample mean and then dividing by the sample standard deviation

5 Results

In this section, we compare the models based on the achieved metrics on the test dataset and consider the obtained confusion matrices. Note that all the confusion matrices we present are normalized such that each row sums up to 1, i.e. each row presents the distribution of classes that the model predicts when the true class is given by the row index.

As we already mentioned, the client mainly wanted to maximise the accuracy, while we also consider the achieved ranked probability score as it quantifies the inherent ordinality of the target variable. In the results we also include a baseline model, which uses the proportions of each class in the test dataset as probability predictions for any observation and it serves as a trivial model that we should be able to beat. Furthermore, we denote models trained on the small dataset as Small and due to the large size of the dataset, the only[20] model trained on the big dataset is the Light Gradient Boosting Machine Big (this is the model we will use to discover potentially new useful regressors).

5.1 Imbalanced dataset

Let us first present the models trained without consideration of the class imbalance. Table 6 reports the achieved ranked probability score for each class and the weighted average with weights given by proportion of each class in the test dataset. Note that the weighted average corresponds to the mean RPS over all observations.

	Light Gradient Boosting Machine Big	Baseline Model Small	Light Gradient Boosting Machine Small	Ordinal Regression Small	Support Vector Machine Small
A	0.311	0.413	**0.358**	0.369	0.405
B	0.093	0.247	**0.127**	0.132	0.154
C	0.040	0.088	0.056	**0.054**	0.077
D	0.029	**0.027**	0.032	0.035	0.041
E	0.060	0.113	0.080	0.097	**0.061**
F	0.081	0.259	0.170	0.203	**0.162**
G	0.071	0.421	**0.293**	0.343	0.295
Weighted Avg	0.042	0.082	**0.060**	0.066	0.067

Table 6: Ranked Probability Score for models with unbalanced dataset. The bold values indicate the best value (minimum) in given row (excluding the Light Gradient Boosting Machine Big model, which achieved the best result in all classes – this is the gradient boosting model trained on the *big* dataset). The Weighted Avg row gives the weighted average of each column, where the weights are the proportions of the given class in the test dataset.

Table 7 reports the achieved accuracy of each model on the test dataset.

[20]due to the aforementioned computational difficulties, we trained only this model on the big dataset since it performed best on the small dataset (and we were not able to train the ordinal regression model due to the same memory limitations as explained earlier and the SVM model does not allow for any feature importance interpretation)

Light Gradient Boosting Machine Big	Baseline Model Small	Light Gradient Boosting Machine Small	Ordinal Regresion Small	Support Vector Machine Small
0.657	0.438	**0.565**	0.533	0.519

Table 7: Accuracy for models with unbalanced dataset. Bold value indicates the best performing model (again excluding Light Gradient Boosting Machine Big).

Tables 8, 9, 10, 11, 12 contain the achieved precision, recall, F1 score and ranked probability score for each class and a summarising weighted average for each of the models.

Figure 1 shows the confusion matrices obtained from each model on the test dataset (note that the confusion matrices are normalised such that each row sums up to 1). Let us first note here that the confusion matrix for a perfect predictor would be the identity matrix. Note that the only model that is able to predict even minority classes with nontrivial accuracy is the Light Gradient Boosting Machine Model Big (Figure 1b). This is not particularly surprising, as this model has access to much more detailed data compared to the other models. For the other models, it is quite noticeable that they mainly predict the majority classes (particularly class D), while the performance on minority classes is quite poor. What is particularly interesting is that the Ordinal Regression Small model (Figure 1d) predicted no B classes and only very few A classes[21].

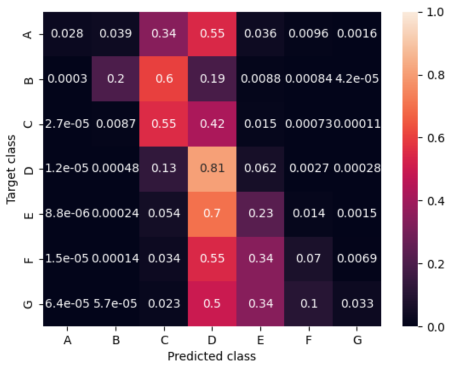
Class	Precision	Recall	F1 Score	Ranked Probability Score
A	0.000	0.000	0.000	0.413
B	0.000	0.000	0.000	0.247
C	0.000	0.000	0.000	0.088
D	0.438	1.000	0.610	0.027
E	0.000	0.000	0.000	0.113
F	0.000	0.000	0.000	0.259
G	0.000	0.000	0.000	0.421
Weighted Avg	0.192	0.438	0.267	0.082

Table 8: Precision, Recall and F1 values for Baseline Model Small with unbalanced dataset.

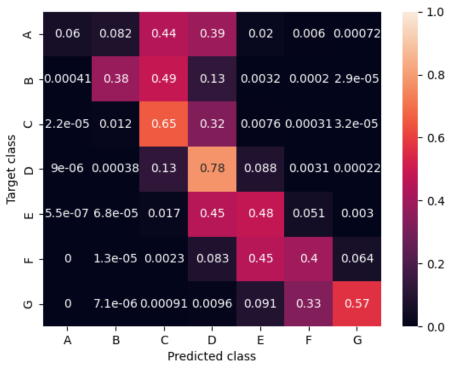
Class	Precision	Recall	F1 Score	Ranked Probability Score
A	0.623	0.060	0.110	0.311
B	0.703	0.380	0.493	0.093
C	0.729	0.655	0.690	0.040
D	0.649	0.780	0.708	0.029
E	0.581	0.478	0.524	0.060
F	0.548	0.401	0.463	0.081
G	0.691	0.567	0.623	0.071
Weighted Avg	0.657	0.657	0.651	0.042

Table 9: Precision, Recall, F1 and ranked probability score for Light Gradient Boosting Machine Big with unbalanced dataset.

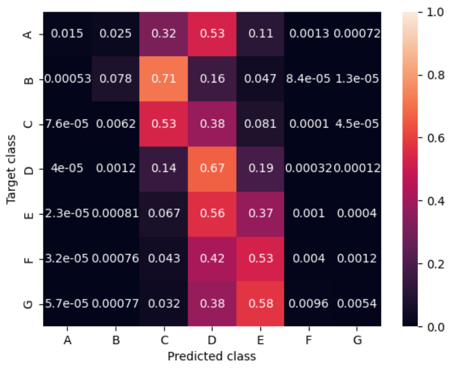
[21] we investigated why that is (if there is maybe a bug in our code) and didn't find a reason. Note that the Support Vector Machine Small model (Figure 1c) also predicts very few classes as A and B



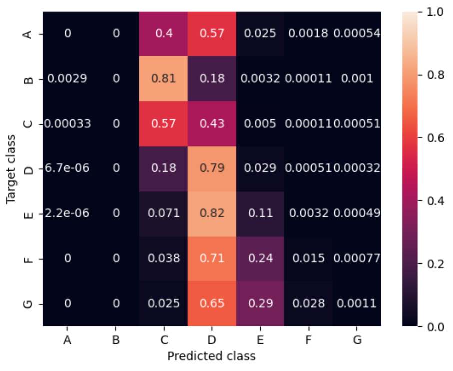
(a) Confusion Matrix for Light Gradient Boosting Machine Small Model.



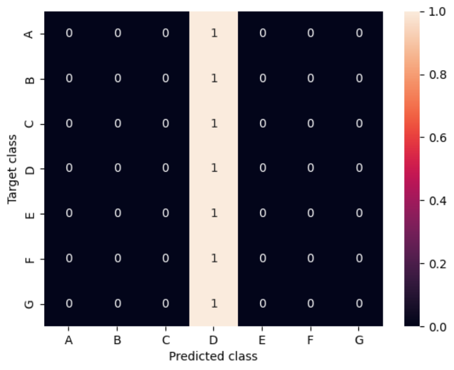
(b) Confusion Matrix for Light Gradient Boosting Machine Model Big.



(c) Confusion Matrix for Support Vector Machine Small Model



(d) Confusion Matrix for Ordinal Regression Small Model.



(e) Confusion Matrix for Baseline Small Model

Figure 1: Confusion Matrices for models with unbalanced dataset.

Class	Precision	Recall	F1 Score	Ranked Probability Score
A	0.398	0.282	0.053	0.358
B	0.630	0.199	0.303	0.127
C	0.660	0.551	0.600	0.056
D	0.549	0.805	0.653	0.032
E	0.442	0.228	0.301	0.080
F	0.370	0.070	0.118	0.170
G	0.383	0.033	0.060	0.293
Weighted Avg	0.553	0.565	0.531	0.060

Table 10: Precision, Recall, F1 and ranked probability score for Light Gradient Boosting Machine Small with unbalanced dataset.

Class	Precision	Recall	F1 Score	Ranked Probability Score
A	0.000	0.000	0.000	0.369
B	0.000	0.000	0.000	0.132
C	0.594	0.569	0.581	0.054
D	0.518	0.789	0.626	0.035
E	0.397	0.107	0.169	0.097
F	0.352	0.015	0.028	0.203
G	0.035	0.001	0.002	0.343
Weighted Avg	0.491	0.533	0.477	0.066

Table 11: Precision, Recall, F1 and ranked probability score for Ordinal Regression Small with unbalanced dataset.

Class	Precision	Recall	F1 Score	Ranked Probability Score
A	0.121	0.015	0.026	0.405
B	0.424	0.078	0.132	0.154
C	0.620	0.532	0.572	0.077
D	0.546	0.668	0.601	0.041
E	0.328	0.372	0.348	0.061
F	0.271	0.004	0.008	0.162
G	0.283	0.005	0.011	0.295
Weighted Avg	0.508	0.519	0.499	0.067

Table 12: Precision, Recall, F1 and ranked probability score Support Vector Machine Small with unbalanced dataset.

We thus conclude that the best model for this task is the Light Gradient Boosting Machine Small with regard to both the accuracy and the ranked probability score and it also scores the best with regard to the weighted average of Precision, weighted average of Recall and weighted average of F1 Score. This is not particularly surprising, as in the last few years, gradient boosting has become the go to model for both regression and classification when faced with tabular data.

5.2 Balanced dataset

For the completeness of this text, in this section, we compare the models trained with adjusted class weights in order to force the model to learn to classify also the minority classes well. Unfortunately, the fact that we give more weight to minority classes means that we lose the ability to classify the majority classes as well as before. In this section, when we talk about the Baseline Small model, we consider that this model represents the proportions from the unbalanced dataset (the model is the same as in the previous section on unbalanced dataset). We include this model instead of the baseline model where the classes would be balanced, as then the probability predictions for each class are uniform and therefore obtaining a prediction by taking the argmax of the probability predictions is ambiguous. We thus consider the baseline model as the benchmark for a model that is trained when the distribution of the data is considered.

Furthermore, we must note that in this section, the most important are the shown confusion matrices. It is of course unsurprising that the model trained with balanced distribution performs worse with regard to the accuracy and ranked probability score compared to the model trained with no class weights, as the test dataset is the same for both the balanced and unbalanced models.

Table 13 shows the achieved ranked probability score for each class and the weighted average (weighted by the proportions of each target class in the test dataset). Note that it is not surprising that the Baseline Model Small performs the best in terms of the weighted average of the ranked probability score, since it uses the underlying distribution of classes, while other models don't. Note that for other classes than the majority D, the other models outperform the Baseline Model quite significantly. It is quite interesting that there are quite significant differences between the Light Gradient Boosting Machine Small model and the Support Vector Machine Small model, while the weighted average of the ranked probability score is the same.

	Light Gradient Boosting Machine Big	Baseline Model Small	Light Gradient Boosting Machine Small	Support Vector Machine Small
A	0.312	0.413	**0.260**	0.303
B	0.096	0.247	**0.088**	0.111
C	0.041	0.088	**0.085**	0.101
D	0.029	**0.027**	0.089	0.083
E	0.061	0.113	0.079	**0.065**
F	0.083	0.259	0.095	**0.094**
G	0.073	0.421	**0.170**	0.177
Weighted Avg	0.043	**0.082**	0.087	0.087

Table 13: Ranked Probability Score for models with balanced dataset.

Light Gradient Boosting Machine Big	Baseline Model Small	Light Gradient Boosting Machine Small	Support Vector Machine Small
0.574	**0.438**	0.422	0.422

Table 14: Accuracy for models with balanced dataset.

Table 14 shows the achieved accuracy for each of the models. It is again unsurprising that Baseline Model Small performs the best in terms of accuracy, as it considers the distribution of the underlying data. The Light Gradient Boosting Machine Small and Support Vector Machine Small perform the same in terms of the achieved accuracy.

Tables 15, 16, 17, 18 show the achieved Precision, Recall, F1 score and ranked probability score for each class and the weighted average (again weighted by the proportions of each target class in the test dataset). Note that in regard to the weighted averages of Precision, Recall and F1 Score, the Light Gradient Boosting Machine small achieves slightly better performance than SVM.

Figure 2 shows the confusion matrices for each of the models (note that the confusion matrices are again normalised such that the rows sum up to 1). Note that the confusion matrices are now much more diagonal, as more focus is given to classifying the minority classes correctly (the baseline model is the same as considered as in the unbalanced section as explained above).

Class	Precision	Recall	F1 Score	Ranked Probability Score
A	0.000	0.000	0.000	0.413
B	0.000	0.000	0.000	0.247
C	0.000	0.000	0.000	0.088
D	0.438	1.000	0.610	0.027
E	0.000	0.000	0.000	0.113
F	0.000	0.000	0.000	0.259
G	0.000	0.000	0.000	0.421
Weighted Avg	0.192	0.438	0.267	0.082

Table 15: Precision, Recall, F1 and ranked probability score Baseline Model Small with balanced dataset.

Class	Precision	Recall	F1 Score	Ranked Probability Score
A	0.063	0.330	0.105	0.187
B	0.273	0.785	0.405	0.038
C	0.666	0.631	0.648	0.046
D	0.698	0.529	0.602	0.058
E	0.460	0.536	0.495	0.059
F	0.351	0.613	0.446	0.050
G	0.478	0.778	0.592	0.033
Weighted Avg	0.615	0.574	0.583	0.043

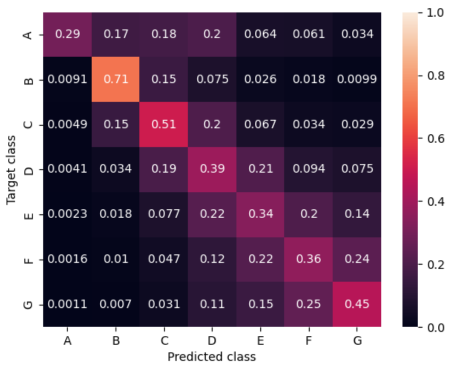
Table 16: Precision, Recall, F1 and ranked probability score Light Gradient Boosting Machine Big with balanced dataset.

Class	Precision	Recall	F1 Score	Ranked Probability Score
A	0.040	0.294	0.070	0.262
B	0.212	0.708	0.326	0.068
C	0.585	0.507	0.543	0.085
D	0.609	0.390	0.476	0.091
E	0.332	0.338	0.335	0.080
F	0.154	0.363	0.217	0.083
G	0.076	0.450	0.130	0.134
Weighted Avg	0.513	0.422	0.449	0.087

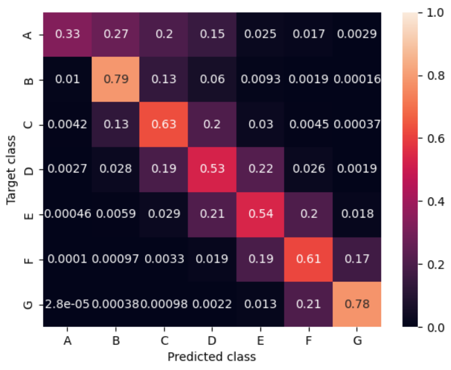
Table 17: Precision, Recall, F1 and ranked probability score Light Gradient Boosting Machine Small with balanced dataset.

Class	Precision	Recall	F1 Score	Ranked Probability Score
A	0.008	0.371	0.015	0.303
B	0.247	0.500	0.330	0.111
C	0.611	0.445	0.515	0.101
D	0.576	0.384	0.461	0.083
E	0.282	0.546	0.372	0.065
F	0.207	0.163	0.182	0.094
G	0.115	0.231	0.153	0.177
Weighted Avg	0.501	0.422	0.440	0.087

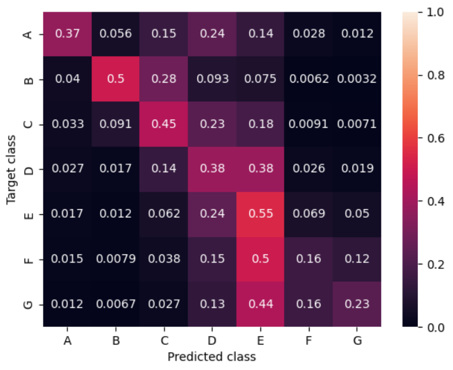
Table 18: Precision, Recall, F1 and ranked probability score Support Vector Machine Small with balanced dataset.



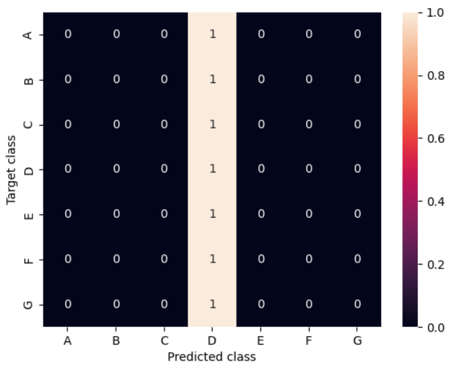
(a) Confusion Matrix for Light Gradient Boosting Machine Small Model.



(b) Confusion Matrix for Light Gradient Boosting Machine Model Big



(c) Confusion Matrix for Support Vector Machine Small Model.



(d) Confusion Matrix for Baseline Small Model

Figure 2: Confusion Matrices for models with balanced dataset.

We thus conclude that if trained with balanced class weights, the SVM Small and Light Gradient Boosting Machine Small models perform the same in regard to the accuracy and ranked probability score, while the Light Gradient Boosting Machine Small outperforms SVM in regard to the Precision, Recall and F1 score. Furthermore, neither of these models beats[22] the baseline model with regard to the ranked probability score and accuracy over the whole dataset.

5.3 Predefined distribution

As discussed in the previous sections, after presenting some intermediate results to the client, it was discussed that the client wants a model that takes into account the distribution of the underlying data (which the balanced version does not) and that the client is working with a distribution that is a little different compared to the one in our training dataset. We were then given a predefined distribution given in Table 5 and asked to train the gradient boosting model (as it performed the best on the unbalanced dataset) with class weights given in such a way, that the proportion of targets given in the training dataset multiplied by the class weights give the predefined distribution (it can be thought of as training the model on a dataset where the proportions of each class are given by the predefined distribution).

We now present the results obtained from training the gradient boosting models using the predefined distribution. Table 19 reports the achieved accuracy of both the big and small model.

Light Gradient Boosting Machine Big	Light Gradient Boosting Machine Small
0.632	0.543

Table 19: Accuracy for models with predefined distribution.

The ranked probability score and other metrics are reported in Tables 20 and 21 and Figures 3a and 3b show the obtained confusion matrices.

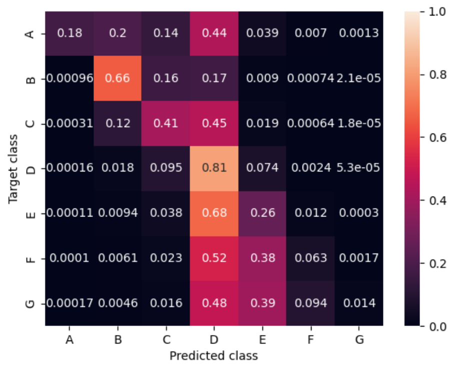
Class	Precision	Recall	F1 Score	Ranked Probability Score
A	0.318	0.177	0.227	0.278
B	0.263	0.658	0.376	0.069
C	0.692	0.412	0.517	0.064
D	0.549	0.810	0.654	0.034
E	0.437	0.260	0.326	0.080
F	0.382	0.063	0.108	0.176
G	0.542	0.013	0.026	0.309
Weighted Avg	0.555	0.543	0.513	0.063

Table 20: Precision, Recall, F1 and ranked probability score for Light Gradient Boosting Machine Small with predefined distribution.

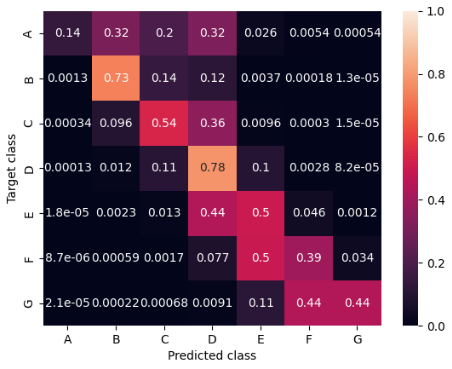
[22]we remind here that the baseline model uses the underlying data distribution while other models don't

Class	Precision	Recall	F1 Score	Ranked Probability Score
A	0.283	0.137	0.185	0.238
B	0.342	0.734	0.467	0.047
C	0.750	0.538	0.626	0.048
D	0.640	0.777	0.702	0.031
E	0.558	0.498	0.526	0.059
F	0.531	0.391	0.450	0.084
G	0.769	0.437	0.557	0.092
Weighted Avg	0.647	0.632	0.628	0.045

Table 21: Precision, Recall, F1 and ranked probability score for Light Gradient Boosting Machine Big with predefined distribution.



(a) Confusion matrix for Light Gradient Boosting Machine Small trained with the predefined distribution.



(b) Confusion matrix for Light Gradient Boosting Machine Big trained with the predefined distribution.

Figure 3: Confusion matrices for models trained with predefined distribution.

We must note again that in this section, the most important are the shown confusion matrices. It is of course unsurprising that the model trained with predefined distribution performs worse with regard to the accuracy and ranked probability score compared to the model trained with no class weights, as the test dataset does not take the predefined distribution into account. We note that the Light Gradient Boosting Machine Small with predefined distribution has accuracy 0.543 and weighted average of ranked probability score 0.063 compared to the accuracy 0.565 and weighted average of ranked probability score 0.060 of Light Gradient Boosting Machine Small with unbalanced dataset (Table 7 and Table 6).

5.4 Potentially useful new regressors

In this section, we aim to discover new potentially useful predictors that the client may collect in order to improve the predictions. To measure which predictors are the most useful, we use the feature importance metric that can be calculated from the gradient boosting model[23] (as we have shown in the previous sections that the gradient boosting model performs the best out of the considered models in terms of accuracy and ranked probability score), which is based on information theory.

Unfortunately, showing how feature importance is calculated explicitly would require us to develop the necessary theory and notation, which is very involved[24] and is outside the scope of this text. We refer the interested reader to [1, Section 2] and [2, Section 2.4.]. To provide at least a very basic non formal idea of how it is calculated for general decision trees for a feature f, it is the information gain (measured using the Gini index) summed over all nodes that use the feature f in the decision tree ensemble. In our case, the Light Gradient Boosting Machine implementation does not use the Gini index but rather the specified objective function[25]. For our purposes, the most important thing is that a higher feature importance means that the feature (variable) is considered more valuable in the model compared to lower importance. We must also note that the obtained feature importance values are not comparable between models trained with different class weights (i.e. the values of feature importances are not comparable between models that were trained with the unbalanced dataset and balanced dataset).

Figures 4, 5, 6, 7, 8 and 9 show the obtained feature importances for the Light Gradient Boosting Machine Big model with unbalanced[26] dataset, Light Gradient Boosting Machine Small model with unbalanced dataset, Light Gradient Boosting Machine Big model with balanced dataset, Light Gradient Boosting Machine Small model with balanced dataset, Light Gradient Boosting Machine Small model with predefined distribution and Light Gradient Boosting Machine Big model with predefined distribution respectively.

For the feature importances in the Light Gradient Boosting Machine Small model with unbalanced dataset, note that in Figure 5, the variables which represent the proportions of the target in given postcode are rated very highly and thus including them in the model is a very good idea. What is also particularly interesting is that for both the models with balanced dataset, the `LOCAL_AUTHORITY_LABEL`, which represents a higher level of regional granularity than `POSTCODE` is also very valuable. Perhaps the same proportions as we calculated for `POSTCODE` could be calculated also for

[23] furthermore, to train the model on the big dataset, we were able to use only the gradient boosting model, as the ordinal regression model failed while training as explained earlier and we were running out of memory when using the SVM model, as it requires one-hotting categorical variables, which produces an extremely large matrix when used on the big dataset (and the SVM model, to the best of our knowledge, does not allow for any such feature importance interpretation anyway)

[24] it would be akin to developing the theory and notation for p-values from scratch

[25] particularly, we refer to LightGBM-feature importance discussion, the objective function in our case is the logloss

[26] when talking about a model with balanced/unbalanced dataset, we mean the model where we use/do not use class weights to train the model as explained earlier

LOCAL_AUTHORITY_LABEL, this could be a significant predictor for the model.

Other potentially useful new regressors can be identified using Figure 4 (in the imbalanced context) and potentially also Figures 6 and 9 (depending on the chosen treatment of class weights). Note that in Figure 4, where the imbalanced dataset is considered, some of the most valuable regressors are related to energy efficiency – it is quite possible, that these are considered when a building receives an energy rating. Obviously, it would be very difficult to obtain this data when the given building doesn't already have an energy rating and acquiring such data would probably be very expensive. It is interesting to note that the proportions of ratings in given postcode are one of the most valuable regressors. Furthermore, the MAIN_FUEL also has a nontrivial contribution – perhaps it could serve as a proxy for the MAINHEAT_ENERGY_EFFICICIENCY and HOT_WATER_ENERGY_EFFICICIENCY (which are probably not easily obtainable by the client). Other notable regressors that could be included in the model are WINDOW_DESCRIPTION and TENURE, which could be acquired from the owner of the property (when he applies for a mortgage) with very little cost.

Furthermore, for the predefined distribution in Figure 9, it is unsurprising that the model considers the B_PROP_POSTCODE as the most significant predictor, as the largest change in the predefined distribution compared to the distribution in our dataset is the proportion of class B. Otherwise the importance of features is not that different to the unbalanced case.

For the completeness of this text, we also include the feature importances from the balanced models in Figures 6 and 7.

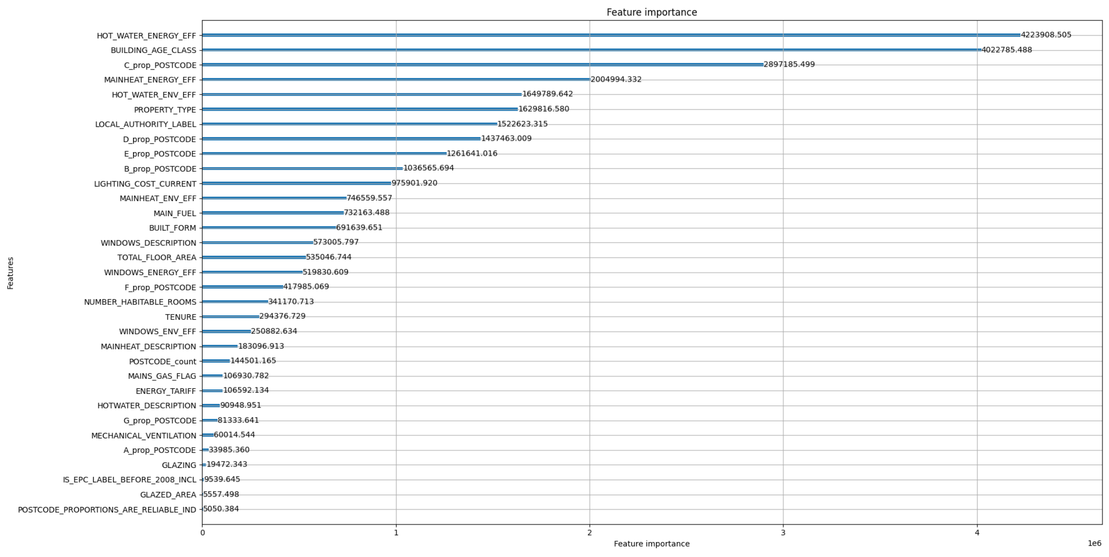


Figure 4: Feature Importance for Light Gradient Boosting Machine Big Model with unbalanced dataset.

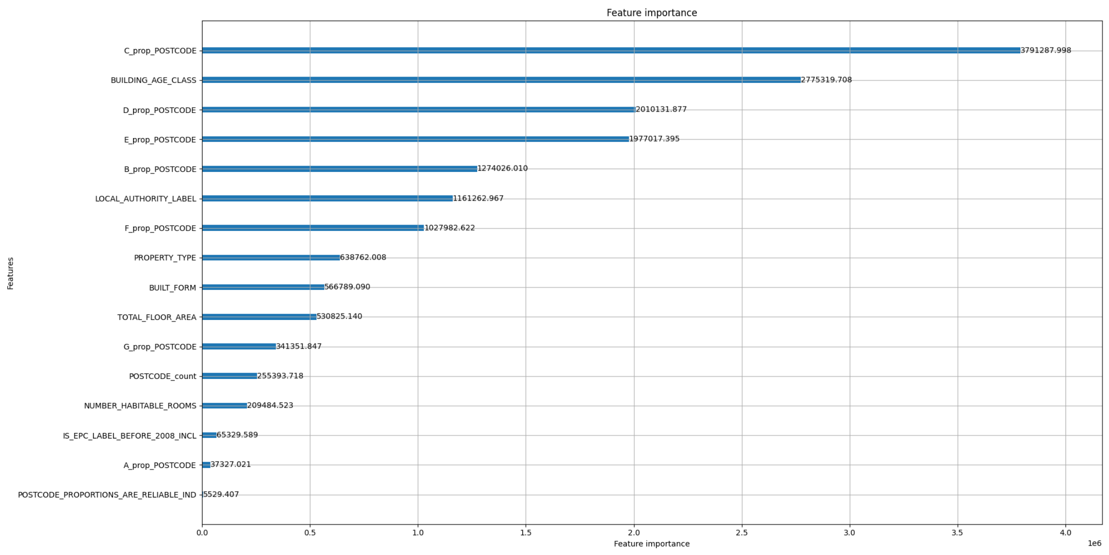


Figure 5: Feature Importance for Light Gradient Boosting Machine Small Model with unbalanced dataset.

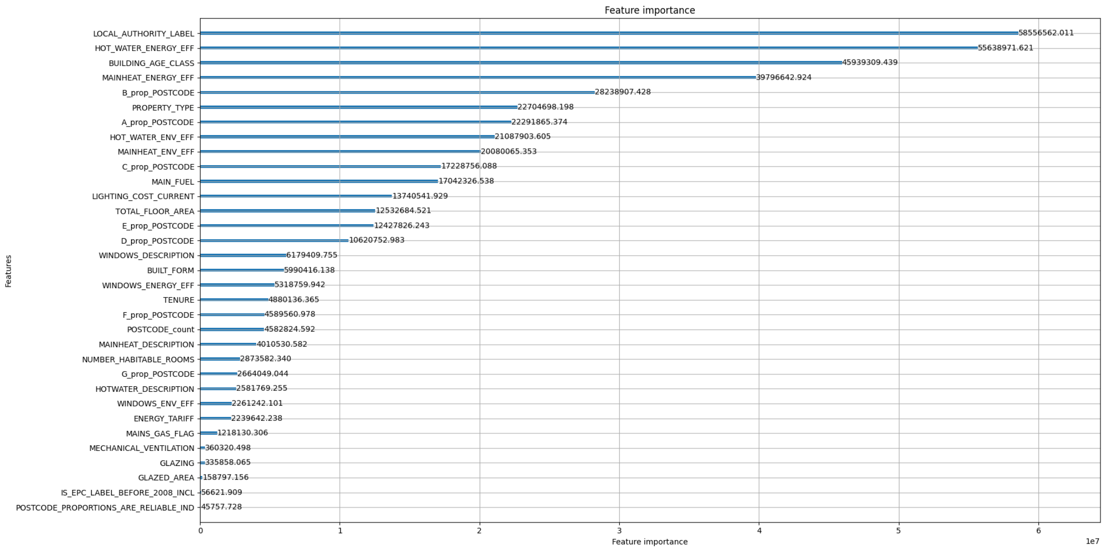


Figure 6: Feature Importance for Light Gradient Boosting Machine Model Big with balanced dataset.

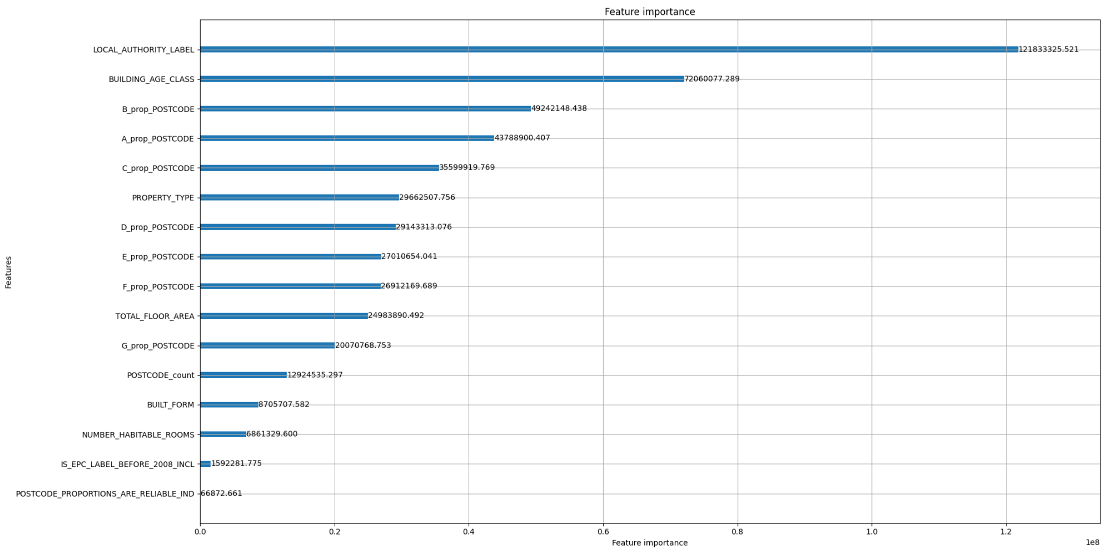


Figure 7: Feature Importance for Light Gradient Boosting Machine Small Model with balanced dataset.

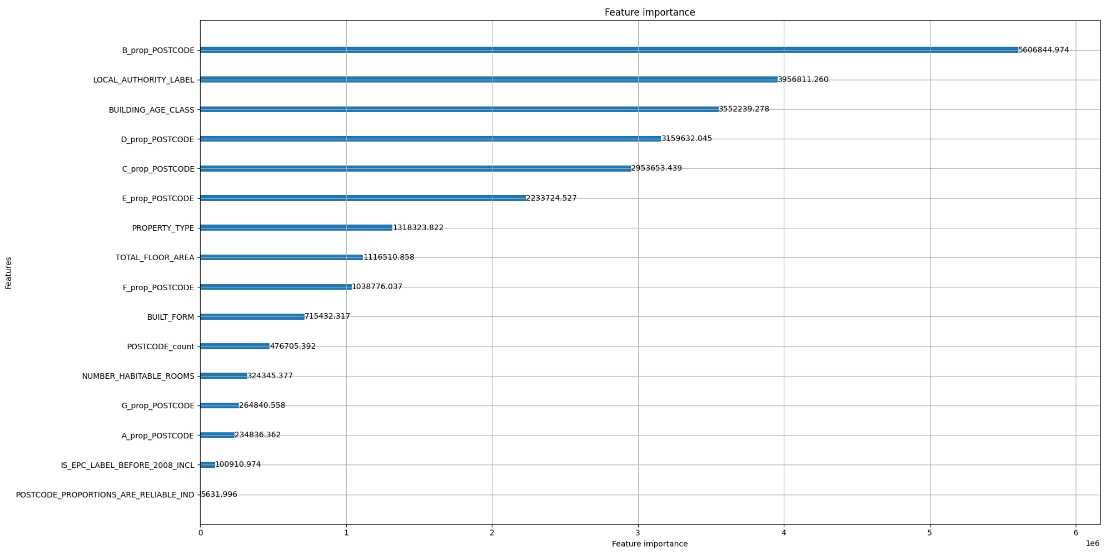


Figure 8: Feature Importance for Light Gradient Boosting Machine Small Model with predefined distribution.

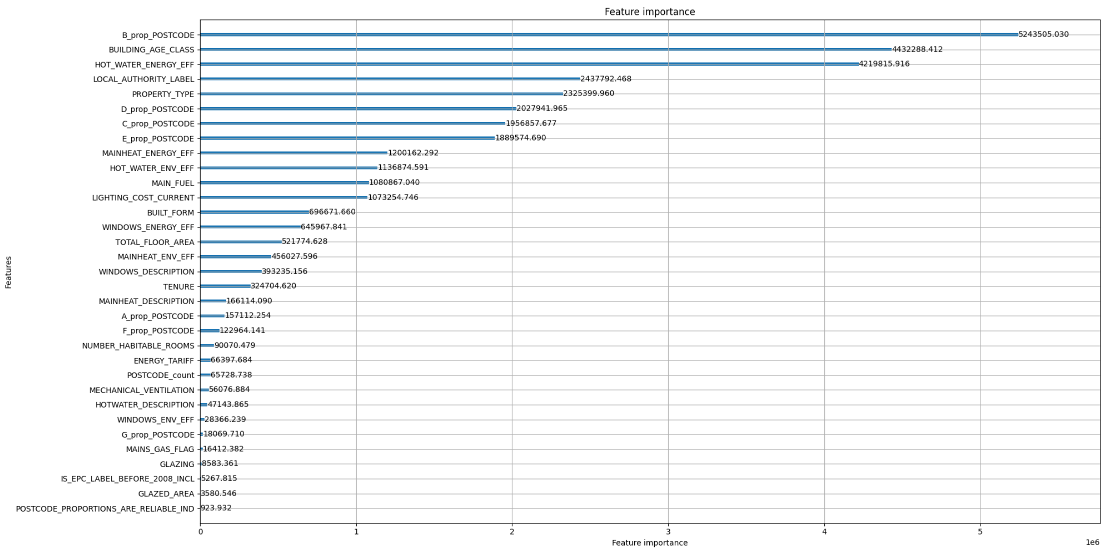


Figure 9: Feature Importance for Light Gradient Boosting Machine Big Model with predefined distribution.

6 Conclusion

In this text, we compared three models that can be used for predicting the energy rating of buildings in the UK. The best performing model turned out to be gradient boosting. Not only did it perform best in terms of the achieved performance, but it was also the easiest to train out of all 3 models, since it does not require any particular data manipulation such as one hotting categorical variables or standardising numerical variables and was able to handle large amounts of data without any problems even on a personal computer.

We did our best to train a model that could be used by the client in practice, which meant that we worked with a very large dataset and had to make decisions (such as not including the `POSTCODE` variable explicitly as categorical in the dataset but rather augmenting the dataset with derived variables) that enabled us to predict the energy rating for as many buildings as possible.

We further used the feature importance metric from the gradient boosting model to discover new potentially useful predictors for the energy rating.

7 Appendix

7.1 Code

The code used for obtaining all results given in this text is available at the authors GitHub[27]. Figure 10 gives a visualisation of the whole codebase.

7.2 Categorical variables

- `CURRENT_ENERGY_RATING`: Energy rating of the building, ‘A’ to ‘G’ (where A is the most energy efficient and G is the least energy efficient),
- `PROPERTY_TYPE`: Describes the type of property such as House, Flat, Maisonette etc. This is the type differentiator for dwellings.
- `BUILT_FORM`: The building type of the Property, one of DETACHED, MID, END, ENCLOSED_END, ENCLOSED_MID, UNKNOWN. See the source code for how these levels are calculated.
- `LOCAL_AUTHORITY_LABEL`: The name of the local authority area in which the building is located.
- `IS_EPC_LABEL_BEFORE_2008_INCL`: Indicator whether energy rating of the building was obtained before 2008.
- `POSTCODE_PROPORTIONS_ARE_RELIABLE_IND`: Indicator whether there are at least 6 buildings in the given postcode
- `MECHANICAL_VENTILATION`: Identifies the type of mechanical ventilation the property has.
- `GLAZED_AREA`: Ranged estimate of the total glazed area of the Habitable Area.
- `WINDOWS_ENV_EFF`: Windows environmental efficiency rating. One of: very good; good; average; poor; very poor. On actual energy certificate shown as one to five star rating.
- `GLAZING`: The type of glazing. From British Fenestration Rating Council or manufacturer declaration, one of; single; double; triple.
- `WINDOWS_ENERGY_EFF`: Energy efficiency rating. One of: very good; good; average; poor; very poor. On actual energy certificate shown as one to five star rating.
- `ENERGY_TARIFF`: Type of electricity tariff for the property, e.g. single.
- `MAINS_GAS_FLAG`: Whether mains gas is available. Yes means that there is a gas meter or a gas-burning appliance in the dwelling. A closed-off gas pipe does not count.

[27] if it is not accessible, please contact the author by email

- `MAIN_FUEL`: The type of fuel used to power the central heating e.g. Gas, Electricity.
- `MAINHEAT_ENERGY_EFF`: Energy efficiency rating. One of: very good; good; average; poor; very poor. On actual energy certificate shown as one to five star rating.
- `MAINHEAT_ENV_EFF`: Environmental efficiency rating. One of: very good; good; average; poor; very poor. On actual energy certificate shown as one to five star rating.
- `HOTWATER_DESCRIPTION`: Overall description of the property feature.
- `HOT_WATER_ENV_EFF`: Environmental efficiency rating. One of: very good; good; average; poor; very poor. On actual energy certificate shown as one to five star rating.
- `TENURE`: Describes the tenure type of the property. One of: Owner-occupied; Rented (social); Rented (private).
- `MAINHEAT_DESCRIPTION`: Overall description of the property feature.
- `HOT_WATER_ENERGY_EFF`: Energy efficiency rating. One of: very good; good; average; poor; very poor. On actual energy certificate shown as one to five star rating.

7.3 Numerical variables

- `NUMBER_HABITABLE_ROOMS`: Habitable rooms include any living room, sitting room, dining room, bedroom, study and similar; and also a non-separated conservatory. A kitchen/diner having a discrete seating area (with space for a table and four chairs) also counts as a habitable room. A non-separated conservatory adds to the habitable room count if it has an internal quality door between it and the dwelling. Excluded from the room count are any room used solely as a kitchen, utility room, bathroom, cloakroom, en-suite accommodation and similar and any hallway, stairs or landing; and also any room not having a window.
- `BUILDING_AGE_CLASS`: Parsed approximate year of construction of a building.
- `A_prop_POSTCODE, ..., G_prop_POSTCODE`: Proportions of current energy ratings by postcode excluding the current property.
- `POSTCODE_count`: Number of buildings in the given postcode.
- `CURRENT_ENERGY_EFFICIENCY`: Based on cost of energy, i.e. energy required for space heating, water heating and lighting [in kWh/year] multiplied by fuel costs. (£/m^2/year where cost is derived from kWh).
- `CO2_EMISSIONS_CURRENT`: CO_2 emissions per year in tonnes/year.

- `LIGHTING_COST_CURRENT`: GBP. Current estimated annual energy costs for lighting the property.
- `HOT_WATER_COST_CURRENT`: GBP. Current estimated annual energy costs for hot water.
- `WINDOWS_DESCRIPTION`: Overall description of the property feature.
- `ENVIRONMENT_IMPACT_CURRENT`: The Environmental Impact Rating. A measure of the property's current impact on the environment in terms of carbon dioxide (CO_2) emissions. The higher the rating the lower the CO_2 emissions. (CO_2 emissions in tonnes / year)
- `ENERGY_CONSUMPTION_CURRENT`: Current estimated total energy consumption for the property in a 12 month period (kWh/m2). Displayed on EPC as the current primary energy use per square metre of floor area.
- `CO2_EMISS_CURR_PER_FLOOR_AREA`: CO_2 emissions per square metre floor area per year in kg/m^2.
- `HEATING_COST_CURRENT`: GBP. Current estimated annual energy costs for heating the property.
- `TOTAL_FLOOR_AREA`: The total useful floor area is the total of all enclosed spaces measured to the internal face of the external walls, i.e. the gross floor area as measured in accordance with the guidance issued from time to time by the Royal Institute of Chartered Surveyors or by a body replacing that institution. (m^2)

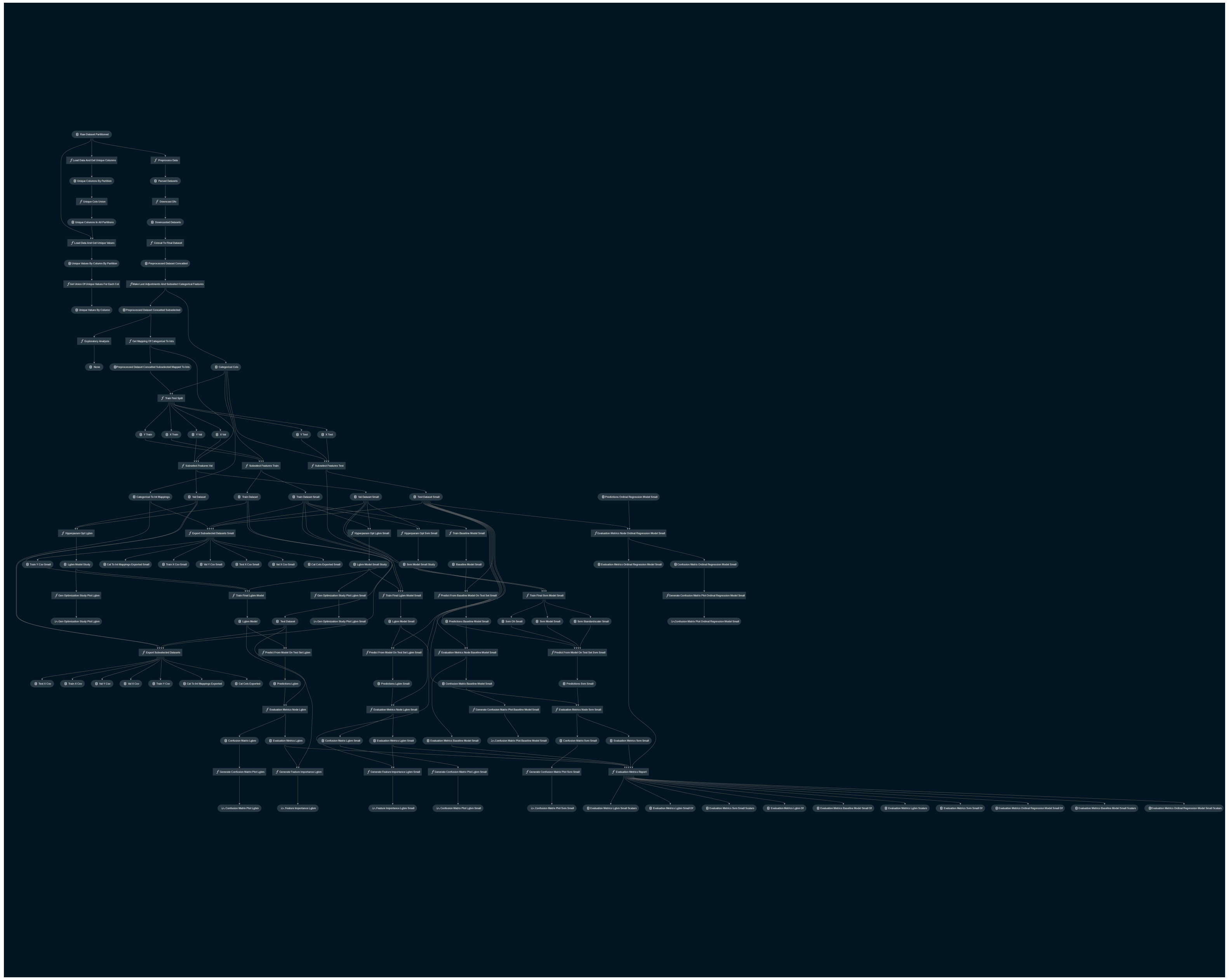


Figure 10: Visualisation of the whole codebase. Zooming in makes the node names readable.

Bibliography

[1] Afek Ilay Adler and Amichai Painsky. Feature importance in gradient boosting trees with cross-validation feature selection. *Entropy*, 24(5), 2022.

[2] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees.* Taylor & Francis, 1984.

[3] Tianqi Chen and Carlos Guestrin. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, aug 2016.

[4] Tomáš Cipra. *Finanční ekonometrie*, volume 30. Ekopress Praha, Czech Republic, 2008.

[5] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[6] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer series in statistics. Springer, 2009.

[7] Masayuki Karasuyama and Ichiro Takeuchi. Nonlinear regularization path for the modified huber loss support vector machines. *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2010.

[8] Guolin Ke, Qi Meng, Thomas Finely, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30 (NIP 2017)*, December 2017.

[9] Allan H. Murphy. A note on the ranked probability score. *Journal of Applied Meteorology and Climatology*, 10(1):155 – 156, 1971.

[10] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490, 2012.

[11] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 08 2002.

[12] Tong Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32(1):56–85, 2004.

[13] Tong Zhang, Fred Damerau, and David Johnson. Text chunking based on a generalization of winnow. *Journal of Machine Learning Research*, 2:615–637, 09 2002.