Discover Estonian Museums with GPT-3

~ Kaggle "Improving the quality of museums data"

TARTU ÜLIKOOL

Daria Eckert - Florian Nebenführ - Till Wenke

Museums Public Portal

BACKGROUND

Estonia has the most museums per 100,000 inhabitants. The Estonian museums information system **MuIS** contains over **4 million items**, however, many objects are not classified ideally.

AIM

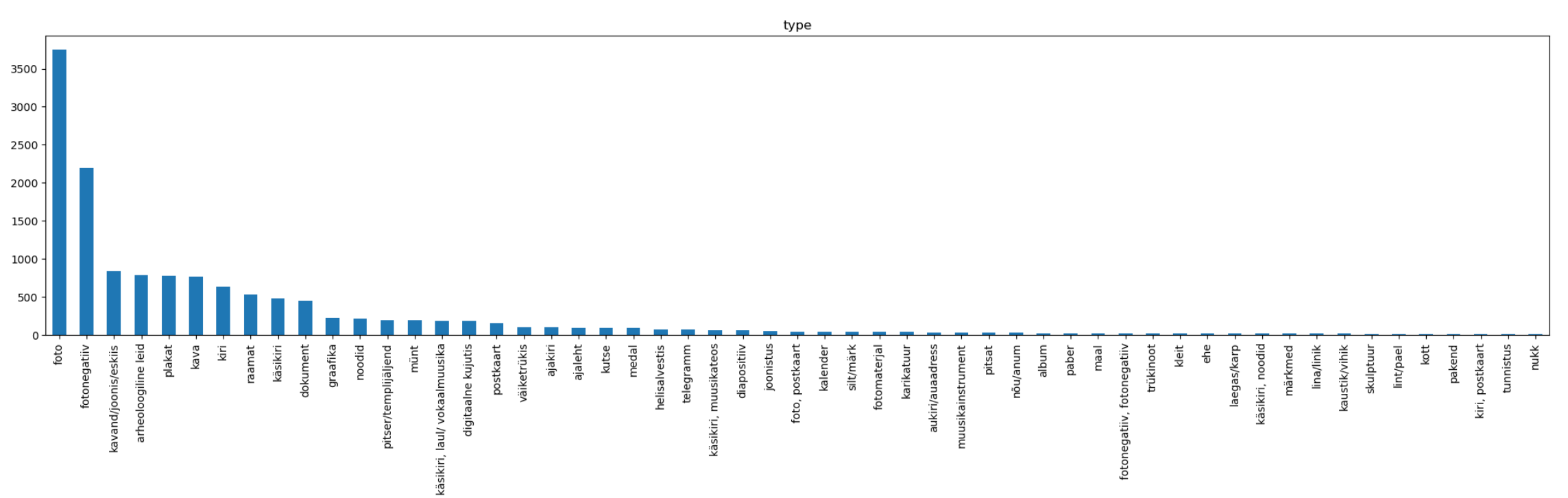
The National Heritage Board wants to "improve Estonian museums information system and preserve Estonia's cultural heritage". To achieve this a model should be built that can **predict** the object **classification type** using a number of descriptive variables. Achieving this would speed up their otherwise manual and time-consuming categorization process. Which in turn would enable a well prepared, accessible database of museum items that could benefit citizens, museum workers and scholars.

DATA DESCRIPTION

TRAINING DATA:

- **14000** Entries
- **34** Descriptive Features
- **55** Categorical Types

The largely text based dataset is made up of several descriptive features and the categorical type of the objects. These features give information about the items; such as its appearance (colour, damages, material), location, timings (start and end times) and museum (museum name, collection numbers). Alongside these are longer descriptive texts that give further clues to the type.



THE CHALLENGES OF THIS DATASET:

- Heavy **imbalance**
- **Sparse** – many NaNs values
- Written in Estonian and Russian
- Large amounts of Text
- Formatting issues

DATA PREPARATION

For this text based dataset the data preparation was crucial. An initial step to understanding the data was to translate the dataset into English using GoogleTranslate via the 'deep translator' library. The next steps were aimed at extracting the most out of the provided data. Through cleaning, reformatting and feature engineering a more uniform dataset was achieved.
For the chosen models the data needed to be numerical. The data was split into a Text dataset and a Tabular dataset (categorical and numerical). The categorical features were converted using hot encoding. The majority of these categorical features had at least 20, up to 3500 different categories, with many of these categories only occurring a handful of times throughout the dataset. To limit the number of additional columns created feature categories were grouped where possible; either by similar meaning (e.g. "colour") or when they occurred less frequently than a set threshold.

Translation · Spliting Dataset · Feature Engineering · Reformatting · Cleaning · Mapping NaNs · Thresholding · Grouping by meaning · Hot Encodeing

MODELLING

MODELS: **1. TabNet** deep neural network for tabular data **2. XGBoostClassifier** implements gradient tree boosting with an ensemble of decision trees **3. Random Forest**

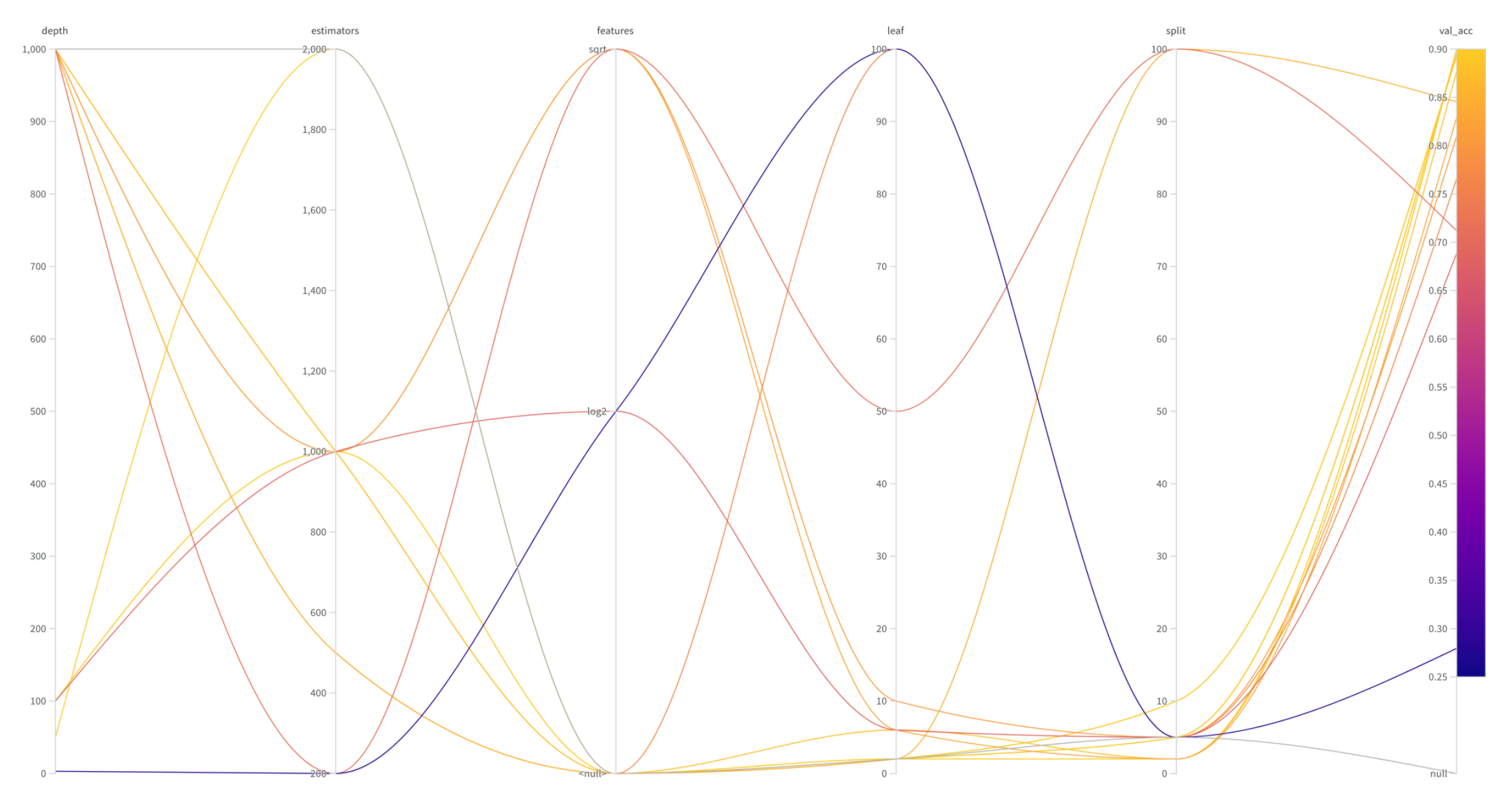
TABULAR DATA:

All three main models were initially trained using a **train:validation** split at a ratio of 70:30 and on both the original and balanced dataset. The dataset has a heavy imbalance with over half of the classification types occurring less than 50 times, therefore to combat this the dataset was **oversampled** using SMOTE. This algorithm augments the data during oversampling, rather than simply copying the data. This is done by choosing the nearest neighbour of the same class and placing the new datapoint on the vector between these two points. After a little experimentation, oversampling all classes with less than 100 instances to 100 instances resulted in the best performance.

To **hyperparameter tune** the RandomForest model, **cross validation** with a fold of 5 and **WandB-Sweeps** were used. WandB's "Bayesian hyperparameter search method" was used to find dependencies within parameters and generate accuracy outcomes computationally quicker than grid or random search. This combined search resulted in the following RandomForest model which was then trained on the tabular data:

RandomForestClassifier(n_estimators=2000, max_depth=100, max_features=None, random_state=0)

Random Forest performed well but default **XGBoostClassifiers** had a better performance but longer training time.


Wandb Sweep

TEXT DATA:

To extract more information from the lengthier text descriptions, OpenAI GPT-3 "Similarity Embeddings" was applied to previously filtered out text data. This AI returns a high dimensional vector to represent the text which can then be converted to features, after which, a model such as RandomForest or **XGBoostClassifier** can be applied.

GPT3

COMBINING DATASETS & MODELS:

To combine the findings from the tabular and the text datasets and combat the uncertainty of the models, a combination of the top models were used (the top 1 - 2 models for the tabular data and the top model for the text data found using GPT-3). This was particularly helpful for the less frequent classification types.
The main approaches:

1. Hard voting: aims to combine models and cancel out their weaknesses.

Here a minimum of 3 models' predictions have to be compared. Alongside these some naïve keyword -> type classification methods were attempted using the text:

1) type name within text [55 pairs]
2) other keywords (indicators) that occur often in the text for one type, e.g., "Puppet" for Doll or "excavations" for Archeological Finds [~140 pairs]
3) using only "save indicators", namely those that only occur together with a certain type significantly often, e.g. "Triumph" for Medal or "denarius" for Coin [6 pairs]

	Type (true type Coin)
TabData RF	Photo
TabData XG	Photo
TextData XG	Coin
Type in text	"don't know"
Majority vote	Photo

1), 2) and 3) yielded little improvement in accuracy when included in hard voting (descending with number of pairs). Trying them in combination with soft voting showed that TabData models already learned those simple relationships.

2. Soft voting: aims to identify and use the model that is most certain in its prediction.

Adds up the class probability of all models and selects the maximum as the predicted class.

	Photo	Coin	Doll	ACC
TabData	0.6	0.3	0.1	91 %
TextData	0.05	0.9	0.05	75 %
SUM	0.65	1.2	0.15	97 %

In many cases the **Soft voting** approach (best model's prediction) **outperformed** that of Hard voting (taking the mode prediction), as different models predicted different types to greater or lesser extents.

TabData	TextData	30 % Test data	Kaggle public	Kaggle private
Balanced XGBoostClassifier	XGBoostClassifier	0.9783	0.9193	0.922
Unbalanced XGBoostClassifier	XGBoostClassifier	0.9616	0.9107	0.9133
Balanced RF	XGBoostClassifier	0.9733	0.9093	0.9123
Unbalanced RF	XGBoostClassifier	0.9869	0.9113	0.9133

RESULTS

96% Average macro Precision

93% Average macro Recall

94% Average macro f1-score

97% Accuracy on the 30% test data

The best performing model was a **soft voting** combination of a **XGBoostClassifier** applied to the vectors created on the Text data and an **XGBoostClassifer** trained on the balanced tabular dataset. On Kaggle the model had an accuracy score of approx. 92% on both the private and public test data. With the help of **GPT-3**, the model was able to predict the less frequent classification types which had as little as 7 occurrences in the training dataset.

TARTU ÜLIKOOL

kaggle