Team Project Report

Forecasting Patient Enrolment for Clinical Trials

presented by
Luka Biedebach
Weiyi Chen
Giang Hoang
Carolin Holtermann
Stefan Sousa

submitted to
sovanta AG
Tommi Kramer & Niklas Frühauf
University of Mannheim

September 2020

Contents

1 Introduction

1.1 Project Background

The first documented clinical trial was conducted in the eighteenth century. It examined whether eating lemons or oranges is a better cure for scurvy with 12 enrolled patients [3]. Clinical trials did not only become more complex in their study design since then, but also very precisely documented. We aim to utilize these large amounts of data to predict the enrollment duration of clinical trials. Our world is facing a countless number of diseases, which raise the need in faster development of medical techniques, vaccine or drugs. However, the clinical trials process is usually expensive, regarding to time and financial terms, especially in the patient enrollment process. The patient enrollment can be a long and costly process for the institution conducting a clinical trial.

In terms of time perspective, a clinical trial should be completed as soon as possible, as competitors' completed clinical trials may render earlier clinical trials irrelevant, even if they have not yet been completed. This is due to the fact that many of the institutions are researching in similar fields, as the highest customer interest and sales are expected for acute or popular topics.

Furthermore, optimizing cost always is a mandatory task for any project to avoid unnecessary annoyance and discouragement. It is possible to estimate the duration and cost of the study based on its research design, but the patient enrollment remains a unknown variable. Several studies tried to model and predict the outcomes and success rates of clinical trials [5], but there are no publications on predicting their enrolment duration yet. This projects stands out from existing research by asking *What impacts the duration of patient enrollment?* and *Can it be approximated with data mining techniques?*. This project aims to answer these questions by gaining insights on clinical trials and the features that are drivers for their duration. Hence, clinical trial records will be collected and preprocessed, to enable the training of a predictive machine learning algorithm with them.

1.2 Clinical Trials

A clinical trial is a study with the goal to acquire new medical knowledge. These studies involve humans either by performing a certain treatment to them, which is called an interventional study or by measuring and observing them over a certain time frame, which is called an observational study [3]. With these interventions or observations, knowledge about diseases, therapies, drugs and devices can be gained. Before a clinical trial can be conducted, it must be reviewed and approved by the relevant national health authority. Pharmaceutical companies are legally obliged to successfully conduct clinical trials before releasing a new drug or device. Clinical studies therefore provide evidence of the safety and efficacy of the respective treatment.

Clinical trials can be initiated by research institutes, pharmaceutical companies, federal institutions or other groups. The studies are commonly conducted by a team of doctors and nurses and take place in hospitals, universities or doctors practices. They can take from a few months up to several decades to be completed. Since every clinical trial requires human participants, the enrollment of them is a important prerequisite for a successful clinical trial. The eligibility criteria defines the requirements someone must fulfill to be a potential participant. This often includes the gender, age and condition of participants. Reasons to participate in a clinical study are the personal ambition to receive a treatment that is not open to the public yet or the participatory remuneration.

A clinical trial consists of four different phases. The earliest phase that includes humans, Phase 1 tests the dosage and safety of a drug. It usually has 20 to 100 participants and takes a couple of month to be completed. Phase 2 examines the interventions efficiacy and possible side effects. In this phase, up to several hundred patients participate. This phase can take up to two years. Phase three focuses on monitoring of adverse reactions with up to 3000 participants, which can take up to 4 years. The last phase, Phase 4 happens after the introduction of the drug or device to the market and has no upper limits in enrolment count or time. We focused on the phases one, two and three in our project, as can be seen in Figure 1

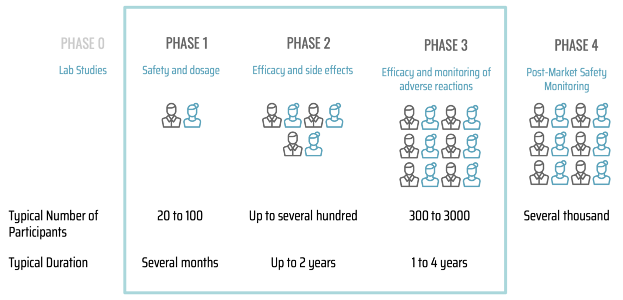


Figure 1: The Size and Timeframe of Trial Phases

1.3 Goals

In this project, we introduce a machine learning model that predicts the enrollment duration of a clinical trial, given a target number of patients and their preconditions, a set of possible locations for trials facilities and different study's parameters. Our mission is to understand the impact of different features on the target variable enrollment duration. We aim to get a substantive understanding of how clinical trials work and which factors determine their enrollment duration. To make the predictions as accurate as possible, we enrich the data set with additional data crawled from various external sources. Information about the trials location, such as population density, age and gender structure or economic information are supposed to model the conditions in which the enrollment process takes place. Also, information about the location facility itself will add valuable information to the data set. Exhaustive preprocessing, model selection and hyperparameter tuning is going to perform the best possible prediction accuracy with the data at hand.

The ability to estimate the enrollment duration of a clinical trial will be very valuable for medical and pharmaceutical institutions. A precise prediction can save costs and reduce the risks risks for these companies due to the improved ability to plan the budget ahead. In addition, an accurate model could also help in developing a recruitment strategy by simulating different study characteristics and the resulting changes in enrollment duration.

2 Data

2.1 Clinical Trials Data Set

The main data source for this project was the publicly accessible clinical trials database provided by the U.S. National Library of Medicine[1]. Each clinical trial record was created by the research institute performing the study. According to Section 801 of the Food and Drug Administration Amendments Act (FDAAA 801) American research institutes were legally obliged to submit information on their studies. Certain trials also need to provide the research results after completing the study.

In total, the database contained more than 350,000 records from 216 countries. Approximately half of these studies were already completed. In the course of the project, we only considered clinical trials that are interventional and test a new drug. Observational studies and studies on surgical procedures and behaviour were excluded. Each record contained the minimum information about the research topic of the study, the study design, the location, the timeframe and the eligibility criteria.

2.2 Data Gathering Process

Retrieving data of the clinical trials through the websites API enabled the work with it in a structured form. The data sets could be filtered and a selection of characteristics could be made before they could be retrieved in JSON format. There is a limitation to a query a maximum of 10,000 full studies as well as a limitation to 20 selected attributes for one HTTP Request Call. To solve this we developed a script that first queries a list of all NCTIDs (unique ID of a study on ClinicalTrials.org) and then systematically retrieves each study individually with the desired attributes.

In order to reduce the amount of data to be retrieved in advance and to retrieve only study records that were relevant to the project objective, several filters were applied during the crawling process. These filters include:

- **OverallStatus:** The status of the clinical trial had to be completed, as the aim of the project was to predict the overall duration of clinical trials under certain influencing factors. Partially completed studies would therefore not provide the required information.

[1] https://clinicaltrials.gov/

- **InterventionType:** Clinical trials are used to test different kinds of medical treatments i.e. device, therapeutic, etc.. However, as we are only interested in the testing of medications as this is of interest for pharmaceutical companies, we filtered for the InterventionType being ”drug”.
- **StudyType:** In the course of the project, we only considered clinical trials that are interventional, which means that a certain treatment is performed to different patients. Observational studies, where patients are measured and observed over time should not be taken into account as they are not concerned with drug tests. Therefore the study records were filtered to only contain interventional studies.
- **Phase:** ”Phase=Phase 2,Phase 3,Phase 4”

2.3 Data Storage/Technology

2.3.1 MongoDB

When selecting the database, the question was whether an SQL database or a NoSQL database would be best for the project. Apart from the fact that everyone in the team was only familiar with SQL databases, the following points favoured a NoSQL database, more precisely the widely used database technology MongoDB:

- Flexible Schema: structure is not statically defined, which means that documents can also be changed in their structure at any time
- Scalability & Performance: it is easy to scale and provides a good performance
- Object-oriented: each document in a collection represents an identity
- JSON-like documents: documents are BSON files, which are very similar to JSON. An advantage is that the responses from API calls come in JSON format and can be directly saved into a collection

2.3.2 Structure of the MongoDB

A database in MongoDB was organized in collections representing groups of related documents. Our database ’clinical-trials’ consists of the following collections:

- ’trials’: contains all queried studies from clinicaltrials.org. Every document represents one clinical trial

- ’country’: contains all country data. Every document represents a country
- ’hospitals’: contains all hospital data. Every document represents an institution
- ’populationDensity’: contains density information for specific coordinates

Because the data is exceeding the limited storage space in the free MongoDB, a second one was created. This data base consists of the following collections:

- ’regional’: contains demographic data on metropolitan regions of the OECD countries.
- ’universities’: contains 12000 universities, which have a world rank that refers to their research quality.
- ’worldrank’: combines the cleaned strings of the hospital and university ranking, that match with LocationFacilities from the trials data

2.4 Data Enrichment

To create a more comprehensive model, the original data set is enriched with data from several different subject areas. This includes data on the demographics of the study locations and data about the hospitals and facilities these trials take place in.

2.4.1 Location data

To leverage our model’s performance, we decided to collect additional data that could provide meaningful information. After a closer look at the clinical database, we realized that the facilities where the study is conducted could play an important role within a clinical trial. Besides, we found a NC data file from Socioeconomic Data and Applications Center (SEDAC), providing an ”estimation of population density for the years 2000, 2005, 2010, 2015, and 2020” [1].

The location data obtained from this file was stored in the MongoDB in a GeoJSON format. This enabled us to query information about the population density. In the beginning we planned using the LocationFacility feature to find out the coordinates of the facilities, provided by Google API. However, many unspecific facility names, such as ”Research center” or ”Drug Research Services” could not be found via Google API. So we decided to use information on the city-level (in other words, the LocationCity feature) as input to produce a new feature called ”LocationDensityPopulation”.

2.4.2 Country data

Clinical trials are becoming more and more globalized. The different health-care environments and ethical requirements of each country make clinical trials in several countries very complex [2]. Most of the trials in our data set were conducted in more than one countries. To get the most up-to-date country data, we made use of four different data sources and merged the retrieved attributes to one complete collection. The data sources used were:

- Worldbank databank: The World Bank API provides access to nearly 16000 time-series indicators, which most of them are available online through tools such as Databank and Open Data website[2].
- Wikidata API : An API exposing data from different wiki sites (i.e. Wikipedia, Wikivoyage, Wiktionary, Wikisource) available using SPARQL requests[3].
- Worldometer: A reference website that provides counters and real-time statistics for diverse topics [4].

In a first step, the country data from the different sources, as described above, was crawled and merged into one complete data frame using unique keys like the country name or the country code. Afterwards the information was merged to the trial data set using different approaches:

- Including the average attribute data from the countries and locations involved in the study.
- Splitting the data set containing one row per country and adding the country specific features to each row.
- Splitting the data set containing one row per city and adding the country as well as city specific features to each row.

A transformer class was created for each approach, so that the merging strategy could be determined later to evaluate the best approach.
In addition, the country data contains information on the worldshare, the proportion of the total population that a particular country represents. This attribute has been used to create an artificial attribute called "worldshareFactor" for each country, which represents a floating point value between 0 and 1 depending on the population density of the country. It thus represents a binning of countries into

[2] https://databank.worldbank.org/home.aspx
[3] www.wikidata.org
[4] www.worldometers.info

countries with very small (worldshareFactor = 0.2) to very large (worldshareFactor = 1.0) population density to approximate the number of patients enrolled in a given country as a subset of the total enrollmentCount.

2.4.3 Regional Data

In addition to the country data, the regional data contains demographic information about the different age groups on the regional level. It was retrieved from the OECD[5], an intergovernmental economic organisation that also provides data about its 37 member countries. The data was clustered in *metropolitan areas*. This includes states as well as cities. Therefore, the data was first matched with the field *LocationCity* and if no match in found, the field *LocationState* was used. To have a higher coverage, a deviation of a Levenshtein Distance of 2 was allowed.

2.4.4 Facility Data

Additional information about the LocationFacility was represented by two different sources: Hospital Data and University Data.
The hospital data was retrieved from two different sources. One source was the Ranking Web of World Hospitals by the spanish research institute Consejo Superior de Investigaciones Científicas (CSIC). This source ranks more than 12000 hospitals all over the world by their research activities. Therefore a hospital with a low rank does better research than a hospital with a high rank. The foundation for this rating is the evaluation of each hospitals online publications. The university data is crawled from CSIC aswell. Their ranking also relies on the facilities research strength. The following features can be retrieved from this source:

- Facility Name
- Country
- Link to Facility Website
- World Rank

The facility data is matched on the field *LocationFacility*. In total, 4046 of the facilities mentioned in the trials data get a world rank assigned. All facilities that cannot be matched, e.g. Universities or Companies, get a value of 0.

[5] stats.oecd.org

2.5 Data Analysis

The clinical trials data set consists of 37613 studies. The data set has already been limited to the query of interventional studies only, and additional filters have been applied to limit the data set to data relevant to the analysis objective. All studies with empty values for the country, facility, condition, study duration and enrollment count are excluded, since this information is crucial for prediction. A further reduction of the studies was made depending on the starting year. Like any science, medical research is developing rapidly. The technologies and enrollment methods used today may not be comparable to the situation 25 years ago. Therefore, studies with a start date before 1990 are excluded. The last completed study in the data set is from 2019.
Each trial has more than 80 features, even though not all of them will be used in the model. These features can be described in categories, as seen in the following table:

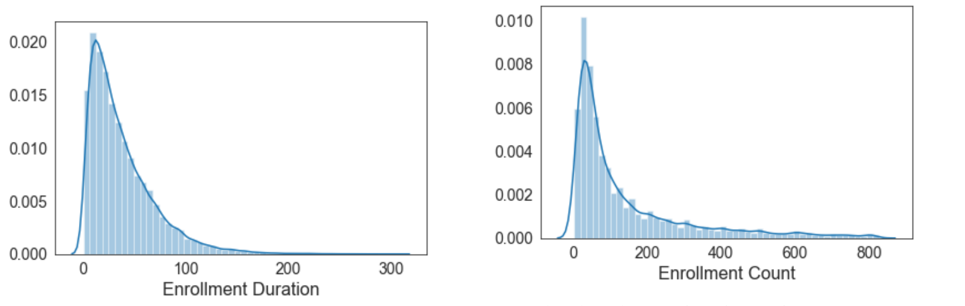
Category	Features
Organisational Information	OrgFullName, OrgClass, StudyType
	LeadSponsorName, LeadSponsorClass, CollaboratorName, CollaboratorClass
	Condition, ConditionBrowseLeafName and ConditionBrowseLeafRelevance
Disease related features	ConditionAncestorId/ConditionAncestorTerm
	ConditionMeshId/ConditionMeshTerm
	EligibilityCriteria, HealthyVolunteers, Gender, StdAge
	DesignAllocation, DesignInterventionModel, DesignPrimaryPurpose
Study Design related features	InterventionName
	IsFDARegulatedDrug, IsFDARegulatedDevice
	enrollmentcount
Study Location	LocationFacility, LocationCity, LocationCountry
Other	ArmGroupLabel, FlowMilestoneType, BaselineGroupDescription, Keywords

Table 2: Feature Categories

The target variable *Study Duration* was initially not included in the data set. Like many other features it was calculated from the original features or added through additionally crawled data. The study duration was derived from the study start date and study completion date. It is measured in months and ranges from 1 to 306 months. On average a clinical trial takes 38.55 months to finish. The enrollment duration itself cannot be directly derived from the available features. Therefore we will temporarily make to distinction between enrollment duration and study duration and get back to this issue later in the project.

2.5.1 Data Distribution

Both, the enrollment count and the enrollment duration, our target variable, show left skewed distributions. There are many small studies with 0 to 100 patients and also a few studies with a very high enrollment count of more than 800. The same applies for the enrollment duration. However, most of the studies have a duration between 0 and 50 months.

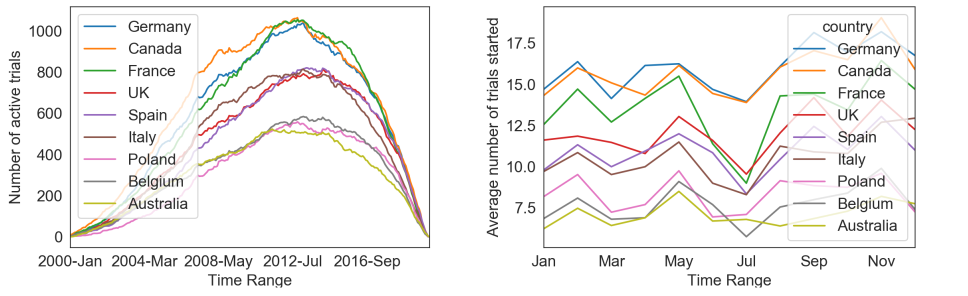


(a) Histogram enrollment duration

(b) Distribution of 95% of the enrollment count

Figure 2: Histograms of important features

Another interesting way to visualize the data is to calculate the number of trials currently run by a country in each month of the time range available. Figure 3a shows this distribution for the 9 countries that conduct the most studies.



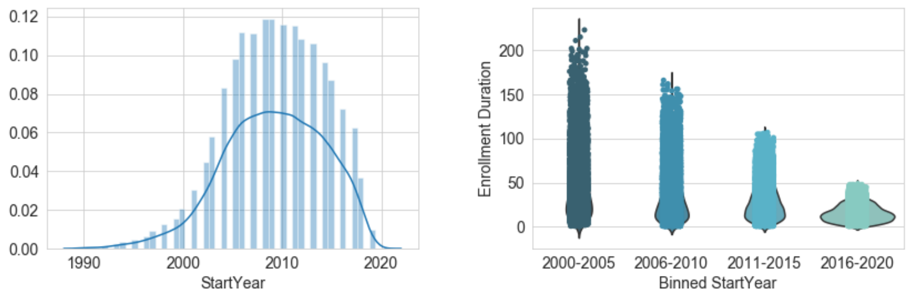
(a) Number of studies a country participates in at a time (without USA)

(b) Average number of trials starting per month

Figure 3: Visualization of the study distribution over countries

For better illustration, the data from the USA are excluded, as the number of studies conducted in the United States is much larger than in the other countries. It can be seen that the number of trials conducted in the different countries follows approximately the same distribution. This indicates that countries are not competing for studies, as all countries follow an upward trend and do not cause a downward trend

for other countries. The downward trend that all countries show when it comes to more recent dates is caused by the fact that only completed trials are included in our data set.
A different analysis can be made when averaging over the number of studies started each year in the different months per country. In Figure 3b we can see, that all countries, start the least studies in June, while most of the studies are started in August and October. For this reason, a feature called ”StartMonth” will be created in the preprocessing pipeline.



(a) Histogram StartYear (b) Boxplot of binned StartYear after 2000

Figure 4: Visualization of the StartYear feature

The distribution of the trials by year is limited by the fact that only completed studies can be considered for the prediction. Therefore, the studies with a recent ”StartYear” are ”naturally filtered” to be short studies only. Figure 4b shows the distribution of Enrollment Duration for countries with a certain start year. The start year is therefore binned into equal width bins of five years. It can be seen that the study duration of the records in our data set almost linearly decreases when the start year of the study increases. For example can trials with a ”StartYear” after 2018 not be longer than 24 months. To avoid any model bias from this effect, all trials with a StartYear after a certain date are excluded. The distribution of the number of studies by ”StartYear” can be seen in Figure 4a. Most of the studies available in our data set started between 2005 and 2015.

2.5.2 Missing Values

There are many features, including the organisation name, organisation class, title, study type, start date, condition and enrollment count, that have no missing values. Therefore, these fields are likely to be mandatory fields when submitting a clinical trial on the website.

All fields with missing values had to be revised individually to decided whether it is reasonable to include them in the model. Therefore, the percentage of missing values was calculated for each feature. When scanning the results it was decided to generally drop all features with a missing percentage above a threshold of 90% from the data set, such as the the feature *BaselineMeasurePopulationDescription*. These features were assumed to not contain enough information to be used in the model. Further the removal of such features created more storage space in our MongoDB that could be used for more explanatory features. If a feature is particularly important for the model, it should be considered to exclude all trials that have a missing value for this feature. For our model, all trials with missing values in the enrollment duration are excluded, as this is our target variable to predict.

For features with a lower percentage of missing values than 90% and not crucial for the model there are different strategies to handle them. See section 3.2.5 for a more detailed description on the transformer that handles the missing values.

2.5.3 Correlations

To understand what important drivers of the enrollment duration are, the correlation of single attributes and duration can be visualized. Categorical features have to be aggregated to see a correlation. We take the median instead of the mean to avoid biased numbers by outliers. Many categorical features show clear tendencies of durations in different values.

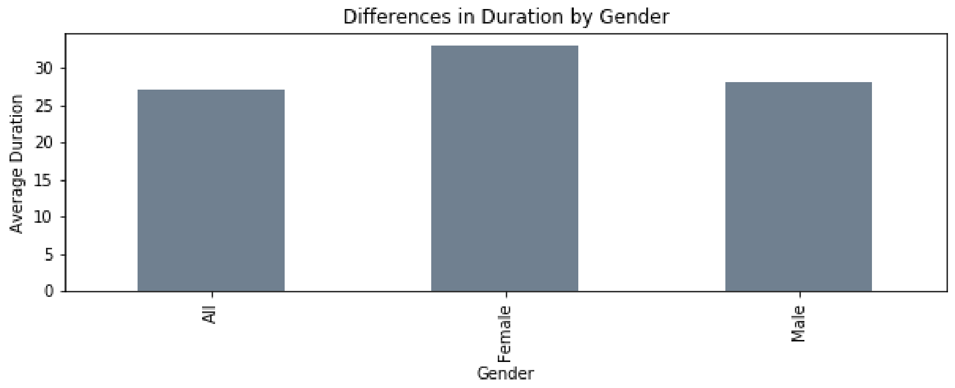


Figure 5: Correlation to Gender

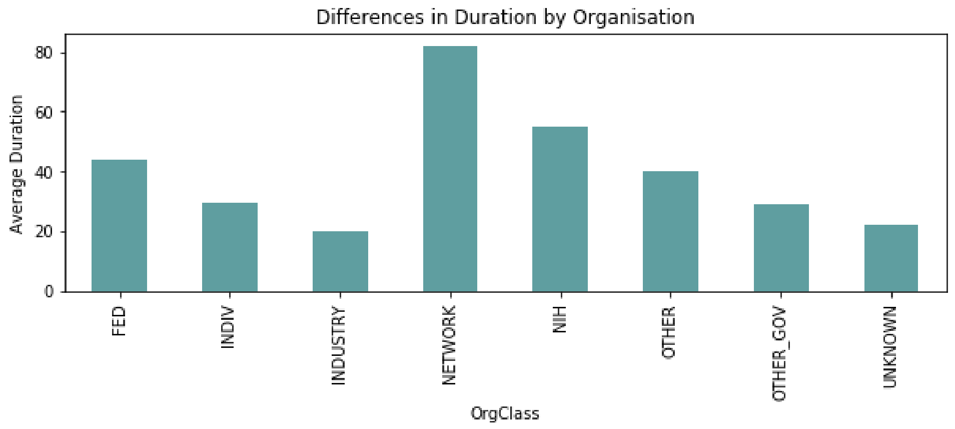


Figure 6: Correlation to OrgClass

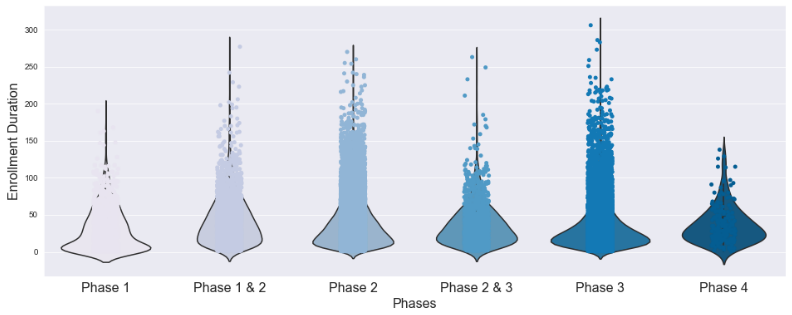


Figure 7: Boxplot of the different phases

If the correlation of one attribute to the target variable duration is too high, it has to be considered whether this attribute is a false predictor. False predictors are attributes that contain data about the target variable and are therefore not realistic in the real application. In this project, ”EndDate” for example would be a false predictor.

Another interesting way to analyse our data is the correlation of each attribute to each other. This shows whether there are highly correlated attributes. Then it has to be decided whether to exclude them from the model, to avoid redundancy. The correlations between all categorical single attributes can be seen in the following correlation matrix:

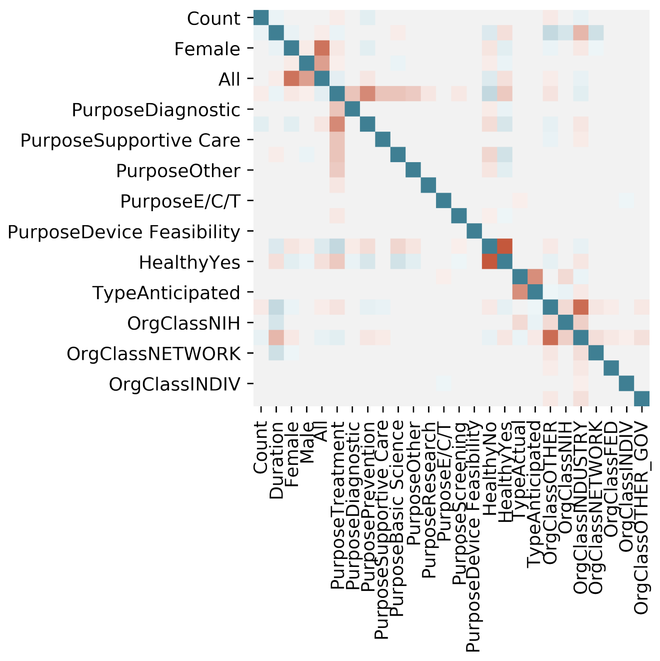


Figure 8: Correlation Matrix

The most important information is the correlation between the explanatory variables and the target variable to identify causal relationships. Since a correlation matrix for categorical values cannot be calculated, it could only be generated after the preprocessing steps were applied and the data set consisted only of numerical values.

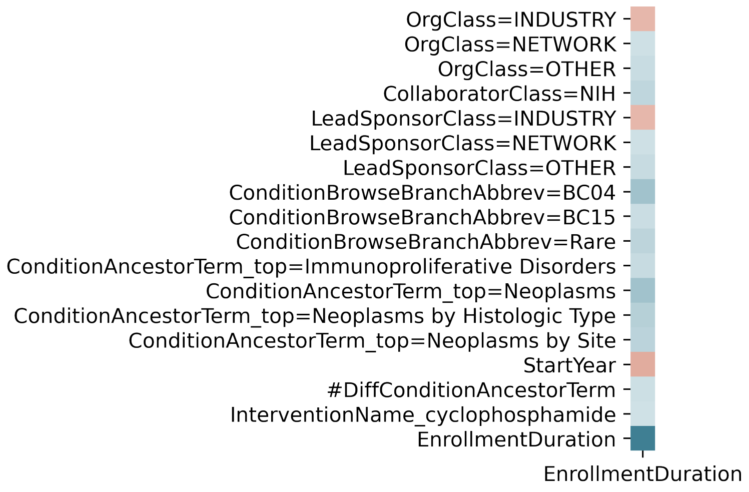


Figure 9: Correlation with Targetvariable

Figure 9 shows all features with a correlation above 0.2 with the target variable. The red squares indicate a negative correlation, while the blue squares indicate a positive correlation. It is noticeable that the condition features make up most of the identified, highly correlated features. Since the condition is a very diverse feature, so far only the 100 most frequently mentioned conditions have been one hot encoded and included in the final data set. This suggests that an improvement of the model could be achieved by including more artificial features derived from the condition. The deviation of the target variable enrollment duration between different conditions can be seen in Figure 10.

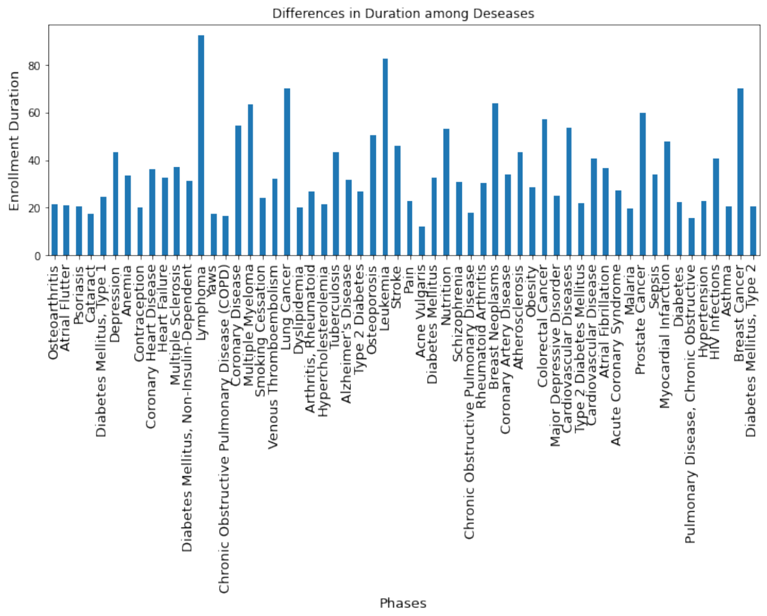


Figure 10: The average enrollment Duration of the trials with the most enrolled Patients

3 Framework

3.1 Overview

We use a framework to find the best prediction model according to defined criteria in a systematic way. The necessity of the framework arose due to the growing complexity in the implementation of the various data mining steps. The advantages include the intuitive extension of preprocessing steps, the traceability and reusability of parameters used in the transformation, the reduction of redundant code and the automatic determination of the best hyperparameters.

Our framework is used to preprocess the raw data in a pipeline and includes an optimization part to select a prediction model and find the best possible hyperparameters for it. The possible parameters also include optional preprocessing steps. The result is an optimal set of transformation steps and a model with optimized parameters.The three main functions of our model can be seen in Figure 11.

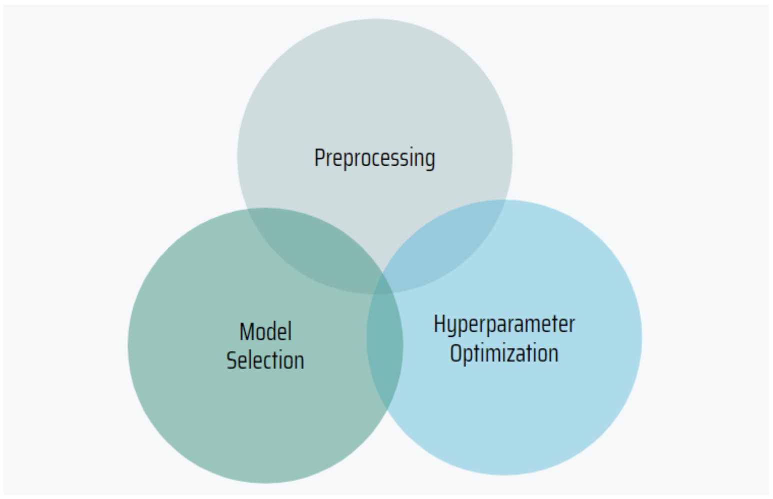


Figure 11: Framework Overview

3.2 Preprocessing

3.2.1 Excluded Data

To ensure high data quality, it was necessary to exclude some records. Some records were already sorted out during the MongoDB query and further inconsistencies were cleaned up before the data was processed further.

The exclusion of entries with inconsistencies was particularly relevant. The following criteria were applied in the exclusion:

- Different number of values among ”LocationFacility”, ”LocationCity” and ”LocationCountry”. Since ClinicalTrials follows a mapping approach through the index of these lists (first entry in ”LocationFacility” belongs to the first entry in ”FacilityCity” and to the first entry in ”FacilityCountry”), the assignment is not clear when these fields have different lengths.

- Relevant fields with references to third pages. These included ”LocationFacility” (e.g. ”For additional information regarding investigative sites for this trial”) and ”EligibilityCriteria” (e.g. ”Please contact site for information”).

Apart from the data that we have excluded in any case, there are still optional exclusion steps implemented in the form of transformers. These include the exclusion of outliers, studies outside a defined time frame, or the exclusion of irrelevant or strongly correlated features.

3.2.2 Pipeline and Transformers

A transformer represents a preprocessing step, which transforms and generates one or multiple new features. Technically speaking, transformers in scikit-learn are classes that inherit from the classes TransformerMixin and BaseEstimator from *sklearn.base*. Transformer objects have a fit method to learn the parameters, a transform method to apply the transformation with the learned parameters, and set_params and get_params to set and call parameters respectively.

A pipeline enables the sequential execution of several preprocessing steps. It contains several transformer objects and is itself an estimator with a fit and transform method. A pipeline requires transformer objects for all steps, except in the last step, where an estimator would also be possible. However, since our framework clearly separates the mandatory preprocessing steps from model execution and optimization steps, no estimator is used in our pipeline.

When calling the fit and transform methods of the pipeline, these methods are executed sequentially on all transformation objects that are inside the pipeline. During the transformation, a transformer receives the output of the previous transformer, processes it itself and returns the processed output.

The following advantages result from the use of pipelines and transformers:

- Pipelines makes it easier to understand the workflow and to combine different steps due to organized, readable code
- The allow to apply the same transformation steps (with the same parameters) on training data and new data, what also allows a meaningful comparison between different models

In addition to the advantages mentioned above, a pipeline also brings challenges. One disadvantage is the lack of traceability in the transformation steps as the pipeline grows in size. Especially when it comes to tracking errors, identifying the responsible transformer and understanding the error can be difficult. Therefore we implemented a transformer called *Debug* (presented later), which shows the intermediate states of the dataframe to ensure traceability.

Additionally, it is important to maintain a uniform data format for a pipeline. The scikit-learn pipeline can handle both, numpy arrays and pandas dataframes. However, they both require different handling, so it was important to agree on a data format, so that the pipeline returns the desired format at the end. Our transformers expect and return a pandas dataframe. To be able to continue using standard transformers from scikit-learn (e.g. Imputer) which return a numpy array, these transformers have been embedded in a custom transformer. This ensured the return of a pandas dataframe.

3.2.3 Types of Transformers

The categorization of transformers follows two approaches: according to their degree of individuality and according to their necessity in the preprocessing part. Categorization according to the degree of individuality:

I. Transformers for a group of features: these transformers apply the transformation to a defined group of features that must go through the same processing steps. For example, one transformer for all numeric attributes.

II. Transformers for specific features: these transformers apply the transformation to a special feature. In this case only this feature goes through these transformation steps. This is the case for domain-specific features such as processing the MeshID.

Categorization according to the necessity in the preprocessing part:

I. Mandatory Transformers: these transformations are always applied, regardless of which model will be executed afterwards.

II. Optional Transformers: as the name suggests, these transformations are optional, as they can be relevant for certain models only.

In addition, there are transformers that play a more organizational role within the pipeline. These are:

- *FeatureSelector*: Returns a subset of the original dataframe with the selected features. This is important to apply Transformers only to a defined group of features.

- *FeatureExcluder*: Removes the original, raw features from the dataframe. This is important to keep only the newly transformed and generated features in the end.

- *FeatureUnion*: required to merge dataframes from different pipelines. This is important after transforming the various features in the respective groups.

- *Debug*: this transformer can be used to trace the intermediate steps between the transformers. The intermediate state can be saved to a file or output to the console. This is particularly important for tracing errors.

3.2.4 Pipeline Example

The next code excerpt shows an example of a pipeline. The different feature groups are processed in separate pipelines. The resulting dataframes are merged with FeatureUnion. The entire dataframe is then transformed further. Finally, the FeatureExcluder takes over the removal of the original features so that the dataframe is ready for the models.

```
pipeline = Pipeline([
    ('features', FeatureUnion([
        ('categoricals_single', Pipeline([
            ('extract', FeatureSelector(CAT_SINGLE_FEATS)),
            ('cat_fill', MissingStringsTransformer(strategy='
                most_frequent')),
            ('single_one_hot_encoding', SingleOneHotEncoder()),
            ('excluder', FeatureExcluder(CAT_SINGLE_FEATS))
        ])),
        ('counting_features', Pipeline([
             ('extract', FeatureSelector(TO_COUNT_FEATS)),
             ('counter', DistinctCounter()),
             ('excluder', FeatureExcluder(TO_COUNT_FEATS))
        ])),
        ('textual_features', Pipeline([
             ('extract', FeatureSelector(TEXTUAL_FEATS1)),
             ('counter', TextualFeatureTransformer( n_keywords =
                 35)),
             ('excluder', FeatureExcluder(TEXTUAL_FEATS1))
        ]))
    ])),
    ('patients_distribution', PatientsDistributionTransformer()),
    ('location_transformation', LocationDataTransformer(
        df_dbcountry, transformer='totalCombine', strategy='
        weighted', mean='worldwide')),
    ('excluder', FeatureExcluder(ALL_FEATURES))
])
```

3.2.5 Transformers

MissingStringsTransformer

Individuality categorical features (e.g. ”Gender”)

Necessity mandatory

Description replaces missing strings using SimpleImputer[6] and returns a pandas dataframe. The standard strategy is 'most_frequent', which uses the most

[6]https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html

frequent value to replace missing values. Alternatively, a defined constant can be used.

MissingValuesTransformer

Individuality numerical features (e.g. ”enrolmentCount”)

Necessity mandatory

Description replaces missing numerical values using different imputation strategies. One can choose between the SimpleImputer[7], the KNNImputer[8] or the IterativeImputer[9]. By default the imputation strategy is to insert the ”mean” value of the other records identified. Alternatively, the median can be used. Further, in case of the KNNImputer, the number of neighbors that are taken into account can be chosen in advance. The Transformer then returns a pandas dataframe free of missing values.

SingleOneHotEncoder

Individuality categorical features, single string values (e.g. ”HealthyVolunteers”)

Necessity mandatory

Description applies One Hot Encoding on features containing single string values. For this purpose DictVectorizer[10] is used, which creates a dummy column for each possible value of a feature. Depending on whether the feature value applies to a record, 1 (applicable) or 0 (not applicable) is assigned.

MultipleOneHotEncoder

Individuality features containing string lists (e.g. ”LeadSponsorClass”)

Necessity mandatory

Description applies One Hot Encoding on features containing lists as values. Follows the same approach as SingleOneHotEncoder, but with the possibility that a record can take multiple values of a feature. MultiLabelBinarizer[11] was used for this purpose.

[7]https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html
[8]https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html
[9]https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html
[10]https://scikit-learn.org/stable/modules/generated/sklearn.feature$_extraction.DictVectorizer.html$
[11]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MultiLabelBinarizer.html

MultipleTopOneHotEncoder

Individuality features containing string lists (e.g. ”Condition””)

Necessity mandatory

Description applies One Hot Encoding on features containing lists as values, but only taking a subset of all possible values. Since some features have a large number of possible values, it was necessary to select these values according to certain criteria. The transformer offers two possible approaches to define top values:

- Strategy ’top’: take the x most common values of the feature, x being the parameter ’top’ to define
- Strategy ’min_value’: take all values above a threshold x, x being the parameter ’min_value’ to define

DistinctCounter

Individuality features containing string lists (e.g. ”Condition” and ”LocationFacility”) and ”EligibilityCriteria”

Necessity mandatory

Description takes over the counting of the different values of a list of a certain feature. For example, the transformer can be used to prepare a new feature with the number of countries involved in the trial. This transformer also handles an exception for the feature ”EligibilityCriteria”, where not the number of different values are counted, but the length of the description. This follows the naive assumption that studies with a longer list of participation requirements and therefore longer descriptions may last longer.

PatientsDistributionTransformer

Individuality applies to ”LocationFacility” and ”LocationCountry”

Necessity mandatory

Description creates two new features by distributing the number of patients in each study among the number of sites and countries involved. The result is the average number of patients per site and country.

ToYearTransformer

Individuality applies to features ”MinimumAge” and ”MaximumAge”

Necessity mandatory

Description standardizes the age fields by converting the different ages into years. For example, the value ”12 months” is converted to ”1 year”.

StartMonthTransformer

Individuality applies to feature ”StartDate”

Necessity mandatory

Description extracts the month from the start date and creates a new column ”StartMonth” in the data set containing the extracted month as float value.

StartYearTransformer

Individuality applies to feature ”StartDate”

Necessity mandatory

Description extracts the year from the start date and creates a new column ”StartYear” in the data set containing the extracted start year.

TextualFeatureTransformer

Individuality applies to textual features (e.g.”OrganisationName”)

Necessity mandatory

Description takes a dataframe containing the textual features as input and applies one hot encoding to keywords after several steps. First, basic NLP preprocessing steps are applied to normalize the sentences and remove stop words. Then the sentences are divided into lists of word tokens that can be iterated and compared. Afterwards the top k keywords that are mentioned most often are extracted, the number being determined by an input parameter. Finally, a new column is created for each of the keywords and a hot encoded (1 if the keyword is present in the sentence, 0 if not).

TimeOutlierRemover

Individuality applies to the whole data set

Necessity optional

Description this transformer takes two parameters as input, which define the time period the data set should cover. This could prove to be advantageous, since the medical and technical possibilities for conducting a study have changed considerably over the years, and so has the required enrolment period for patients. Therefore, this transformer can be used to exclude data sets before a certain date.

enrolmentCountOutlierRemover

Individuality applies to the feature ”enrolmentCount”

Necessity optional

Description takes the outlier identification strategy as an input (i.e. ”IQR” or ”MAD”) and removes records whose enrolment count lies outside the range of what is assumed to be the ”normal”.

LocationDataTransformer

Individuality applies to the features ”LocationCity” and ”LocationCountry”

Necessity mandatory

Description applies to the location features and includes three transformers with different strategies to merge the country data to the main trials data set. The different combination strategies are: ”totalCombine”, ”perCountry” and ”perCity”. The input parameter ”transformer” defines the strategy that should be applied.

totalCombine: this strategy adds information on the countries and cities involved in the study. The lowest value, the highest value and the average value of all country attributes are added as new features. At the city level, the lowest value, highest value and average value of the population density of all cities involved in the study are added.

perCountry: the main goal of this strategy is to split the original data set to contain one row per participating country in one study record. Although there are then multiple rows for the same study, they still share the same unique identifier ”NCTId”. Also, all other features besides the country features are simply copied for each row. This approach is applied in order to be able to fully include the country information available.

In order to be able to estimate how many people were enrolled in a specific country, in a first step an artifical feature ”worldshareFactor” is created additionally to the country dataset. Based on the worldshare each country has, a ”worldshareFactor” between 0 and 1 is assigned. Furthermore, the trials data set is enhanced with an artificial feature ”DistinctCountries” which is based on the ”LocationCountry” feature and only contains a set of the distinct countries mentioned in this feature.

Afterwards, the trials data set is split into multiple rows, one row per country, based on the ”LocationCountry” feature. If one country runs the study in several locations in different cities, the features ”LocationCity”, ”LocationState”, ”LocationFacility” and ”LocationCity” are split up in smaller lists containing the facilties involved in the respective country. Furter, a new feature ”facilities” is calculated, which describes the number of facilities contributing to the study in each country. In a next step, the available country data is merged using the country name as join key. Then, the total enrolmentCount is split per country using the number of locations facilities that contribute to the study per country as well as the worldshareFactor of each country.

perCity: follows a similar approach as perCountry strategy, but divides a study into multiple rows, equivalent to the list of cities in the study.

WorldRankTransformer

Individuality applies only for the special feature ”LocationFacility”

Necessity optional

Description adds a new column called ”AvgWorldRank” to the data set, which describes the average of the available world rankings of the participating institutions. The transformer takes as input a previously created data set containing the world rankings for all institutions present in the trials data set. This dataset was compiled in advance based on the distance to the facilities mentioned in the experimental dataset. The transformer then scans the trial dataset and inserts the world rank data for each facility involved in the study, if available. The mean value of all WorldRanks found is then calculated and added as a new feature.

RegionalAgeTransformer

Individuality creates a new feature from the location data and miminum and maximum age

Necessity optional

Description takes the regional demographic data from OECD. This data contains absolute numbers of citizens in the age groups ”Youth”, ”Working” and ”Elderly”. From the fields minimum and maximum age, a helper column is created that represents the included age groups in an array. The transformer cleans the strings from ”LocationState”, ”LocationCity” from the raw dataframe and ”Region” from the regional data. Because the regional data refers to metropolitan regions, states and cities are included. The transformers first checks whether the city is included. If so, the transformer refers to the helper column and matches the respective absolute population numbers. If the city cannot is not included, the repeats the same process on the state level. Finally, the regional population of the included age groups are summed up.

MeshIDTransformer

Individuality applies only for the special feature ”ConditionMeshID”

Necessity mandatory

Description transforms the Medical Subject Headings ID into an integer. Removes the ”D” in the beginning of the ID and converts it from string into integer. Since some trials have several (mainly 1-3) Mesh ID’s saved in an array, a column for 1st, 2nd and 3rd MeshID are created.

FeatureSelectorTransformer

Individuality applies to the whole data set

Necessity only necessary for models where performance can be improved by using a subset of functions.

Description selects the most important features by comparing feature characteristics to a certain threshold[12] and calculating the feature importance based on lgbm.

[12]https://towardsdatascience.com/a-feature-selection-tool-for-machine-learning-in-python-b64dd23710f0

I. Single unique value: Delete features that only contain one value. And thus do not provide any useful information.

II. Collinearity: Delete features that show a correlation above a defined threshold with other features.

III. Zero feature importance: Delete features that have a feature importance of 0.0 according to the gradient boosting machine (lgbm).

IV. Low feature importance: Delete features that show a feature importance below a defined threshold according to the gradient boosting machine (lgbm).

These thresholds for highly correlated features as well as features with low importance need to be defined as input parameters in advance.

LabelEncoder

Individuality applies to our target ”enrolmentDuration”

Necessity mandatory for classification

Description to turn a regression problem into a classification problem, this transformer converts the target into classes. The new label for each class is its central value. There are two strategies for creating the classes:

- equal-interval binning: each class has the same time length, i.e. duration of 6 months
- equal-frequency binning: same number of trials in each class

  However, equal-interval binning has the disadvantage that the classes are imbalanced, which would require an additional step to balance the differences. For this reason, equal-frequency binning is the default.

TargetConditionTransformer

Individuality applies only for the special feature ”Condition”

Necessity mandatory for modified pipeline

Description transforms the condition list into 2 separate columns. As the feature condition mainly contains 2 or less values in a list. Columns of condition1 and condition2 are created.

3.2.6 The Final Dataframes

When applying the preprocessing pipeline on the raw dataframe we get the final dataframe that is used for model selection and hyperparameter tuning. This has no categorical or list features, no missing values and contains many additionally created features to enhance the prediction. Applying all transformers to the data set it changes its dimensions. If the outlier remover is applied, the height of the data set decreases from 30973 records to 30360, due to the removal of inconsistencies. Further removals will happen in the optimization process if a outlier remover is applied. The one hot encoding and feature engineering increase the width of the dataframe from 55 columns to 1081. This is a quite large number of columns and can negatively impact the model.

To enhance the performance, we want to avoid the curse of dimensionality. In the section of hyperopt, we use target encoding or categorical encoding for single categorical value features. Asides from the SingleOneHotTransformer, we apply the pipeline as usual to get the single categorical features in their original state. They will then, be transformed with target encoding or categorical encoding in the hyperparameter optimization. Also the field *Condition* will be used in the target encoding. It is a list field, which cannot be used for target encoding but more than 90% of the trials only have one or two conditions in this list. We split the list of condition into *Condition1* and *Condition2* to be able to encode it with the target encoder. The final dataframe for this approach has in total 957 columns.

3.3 Model Selection and Hyperparameter Optimization

3.3.1 Gerneral settings for hyperopt

Hyperopt is a Python library for Sequential Model-Based Optimization (SMBO) used for hyperparameter optimization. Most models expect configuration variables called hyperparameters, which allow to adjust the way the model runs. These parameters determine how the algorithm works and can be used.

Hyperparameters have to be defined prior to training, which means that it is required to specify a set of parameters beforehand. In order to systematically try different combinations of hyperparameter values, we used the approach described in ”Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms” by James Bergstra, Dan Yamins and David D. Cox [4].

The following four major steps describe how hyperopt works:

Step 1: Define the objective function

The objective function is used to evaluate the different sets of hyperparameter values. It returns loss values making a comparison possible, and determines which configuration is best according to the loss value. In the both regression and classification, we implemented MAE as our loss function. To clarify, in the classification models, rather than use F1-score or precision, we applied a customized label transformer on it. To begin with, we transfer numerical targets into new labels by applying a customized label encoder which we have two options. The first option is the equal-interval approach which created bins with the same amount of data, and the other option is the equal-frequency which created bins with the same period. After bins were created, we used each bin's central point as the new label for all bin's samples.

Step 2: Define a configuration space

A configuration space must be defined for each model that consists of the hyperparameters and the possible values. This can be, for example, a continuous value range or a selection of discrete values. For cases in which a certain combination of values is not possible or makes no sense, it is possible to define optional variables. In our case, we defined four different optional transformers which are time outlier remover, enrolment count outlier remover, preprocessing and scaler functions.

Step 3: Choose a search algorithm

The search algorithm is responsible for selecting the combinations of values to be tested. Because not all possible combinations of hyperparameter values can be selected, this algorithm is responsible for finding the most meaningful combinations from the configuration space. In this process the objective function is taken into account when searching for the best parameters' combination.

Step 4: Run the optimization driver (fmin)

With the fmin function the hyperparamter optimization is executed, i.e. the search for the best hyperparameter values is started. fmin expects the function to be optimized and the defined configuration space as parameters. Optionally, an instance of the type Trials can be passed to the function.

Fmin returns by default the best result from the tried combinations. To store the remaining results and the respective hyperparameter values, a object of type Trials can be passed. The simple implementation of Trials is based on Python lists and dictionaries and can be output at runtime.

3.3.2 General Guidelines for Hyperopt

For the sake of not leaking information in the model, we splitted the data beforehand. Since our data set consisted of clinical trials conducted in a range of 22 years, the most correct way to split the data is a time series split. This split makes sure that no future data was used to predict the duration past trials. We split data into 3 divisions: training data (72%), test data (18%), and holdout set (10%). We used a holdout set for the hyperparameter optimization to prevent an overfitting of the parameters to the test set.

In the implementation of hyperopt, there are customized functions that chose the best parameters with personal settings. As mentioned in *section 3.2.5*, there are transformers to remove outliers for time and enrolment count. Apart from that, there are preprocessing functions regarding feature selections, PCA, and different scalers to apply.

We use random forest, XGBoost, XGBoostRF, and lightGBM for the prediction in regression models. On the other hand, we added basic models such as naive Bayes, support vector machine, and k-nearest neighbors in classification models. Inside the hyperopt, we included 4 optional transformers, which were time outlier remover, enrolment count outlier remover, preprocessing and different kinds of scalers. The hyperopt model could decide which option returns the best value with fmin function. These customized parameters needed to be deleted before being put into the model. The feature dimension of our output data was originally 1080, which might cause the curse of dimensionality.

The preprocessing option contained feature subset selection and principal component analysis which made the dimension lower. The modified hyperopt which can handle target encoding, we further used the target output from pipeline for target encoding and categorical encoding. The original data dimensional was then cut down to 1055. In the setting of modified hyperopt, we put target data encoder beforehand so the model can further process numerical value and thus do feature selection, PCA, or scaling on it. Since target encoders does not change how the original categorical label functions in categorical encoding, it is fine to put target encoding before categorical encoding. Categorical encoding can then change numerical values back to categorical values.
As the model was completed, we used fit train data to predict the label of test data.

3.3.3 Parallelizing Evaluations During Search via MongoDB

In order to be able to execute several hyperparameter searches asynchronously and in parallel, the HyperOpt module offers a MongoTrials implementation in addition to the Trials[13]. This made it possible to improve the performance by allowing hyperparameter tuning to be executed on multiple devices.

For using the parallelized approach, the Trials instance of the fmin function had to be replaced by a MongoTrials instance. While the Trials instance stores the results of the optimization runs only at runtime, the MongoTrials object uses a MongoDB instance to persist the results. For this, the MongoTrials instance utilizes a local mongod process as input and the information about the database where the optimization results should be stored in.

Additionally, a hyperopt-mongo-worker must be started when the fmin function is executed with MongoTrials. This worker evaluates the target function with randomly chosen parameter settings and writes the results back to the defined MongoDB.

3.3.4 Workaround: Synchronisation of HyperOpt Results

The use of MongoTrials to store the Hyperopt results in a MongoDB has the technical limitation that the communication is only possible with a local MongoDB instance. To synchronize the results, which are calculated decentrally on the computers, we had to implement a workaround.

[13]https://github.com/hyperopt/hyperopt/wiki/Parallelizing-Evaluations-During-Search-via-MongoDB

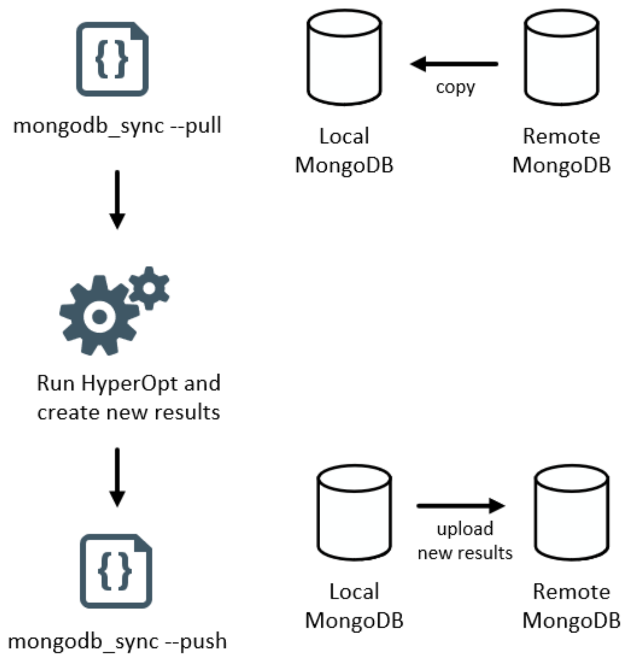


Figure 12: MongoDB Syncronisation

The process starts with the execution of the script with the argument ”–pull”. During this process the local database ”hyperopt” is overwritten by the remote database in the cloud also named ”hyperopt”. After this process the local database is synchronized.

HyperOpt can then be executed locally. The new results are written to the local database and already executed hyperparameter combinations are skipped.

Finally, the script is executed again, but this time with the ”–push” argument. The new results are identified and uploaded to the remote database. The locally assigned ID is removed from the results so that the remote database can assign new document IDs and no conflict occurs. The remote database is now synchronized.

3.3.5 Steps needed to run HyperOpt in parallel

To run the HyperOpt in parallel jobs, several steps need to be carried out[14]:

[14] http://hyperopt.github.io/hyperopt/scaleout/mongodb/

I. Import the pipeline output from the preprocessing as a .csv file.

II. Pull the previously run jobs from the remote MongoDB to your local MongoDB by running the command *python mongodb_sync.py pull*. This will run the pull process of the python file *mongodb_sync.py*.

III. Download the HyperOpt file *HyperOpt and Models.ipynb* and run it locally on your device.

IV. In the fmin function, instead of the Trials() object use the MongoTrials() object and execute the cell.

V. In your command line execute *hyperopt-mongo-worker –mongo=localhost:27017/hyperopt –poll-interval=0.1* to activate a mongo worker that polls and runs the optimization jobs.

VI. When the hyper opt process is done, write the results in the MongoDB with the –push command.

3.3.6 Decisions on parameters

In general, the decisions are made in a regression model, in different kind of models, the decisions can differ. If we want further improvement on the hyperopt parameter tuning, it should be better to make decisions based on what model we are using. We provided four different hyperopt models in our task.

I. Before, lightGBM with boosting type 'goss' resulted in very large values, which in return, would make the model avoid choosing lightGBM. We studied the settings all over, there were alias names existing in lightGBM model. The bagging_fraction should be changed to the parameter name subsample, so that the new model can read.

II. Before, random Forest resulted in overfitting and took too much CPU and runtime, we tried to reduce the value of n_estimators, max_depth, max_leaf_nodes, and max_features in order to cut down the trees

III. Before, PCA's n_component was set to choose between the dimensions, it did not result in good outcome, we then set n_component to None. Which means the n_component will take value from min(n_samples, n_features).

3.3.7 Regression Models

Settings for the Regression Model
There are three regression models we included in the optimization framework to choose from.

I. XGBoost: a ensemble technique, that uses gradient boosting, e. g. learning the residuals of previous models and adds new models. It uses the gradient descend algorithm to minimize the loss of its sequentially added models.

II. LightGBM: a esemble technique, that uses tree based learning. In contrast to other tree based learning algorithmn it grows the trees vertically, e.g. by its leaves.

III. Random Forest: a bagging technique, that randomly creates a number of uncorrelated decision trees. Finally, the prediction is made with a majority vote of these trees.

Each of them requires a individual set of parameters.

Model Selection
Hyperparameter Optimization showed that lightGBM is the best performing model for our data set. In classification lightGBM has the lowest MAE as well. You can see the minimum loss of each algorithm in Figure 14. The hyperparameter optimization chose to use the optional transformer *TimeOutlierRemover()* to exclude all trials before 2005. All other optional transformers were not used.

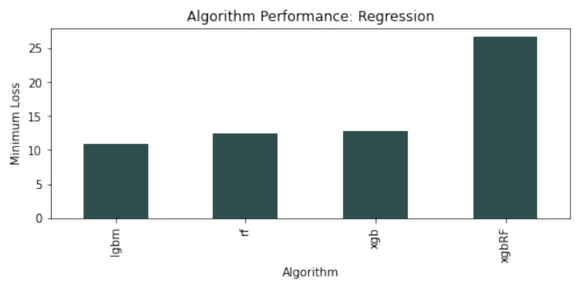


Figure 13: Algorithm Performance in Regression

Results

The hyperopt chose the boosting type ”dart” as the best value. We got the best MAE 10.86 in the validation set was from the modified regression model with the best value on test set MAE 6.73. All chosen parameters can be found in Table. 5 in the appendix.

3.3.8 Classification Models

Settings

There are five classification models we included in the optimization framework to choose from. The objective function was modified. It takes the median values for each label, so that we can use MAE as evaluation function.

I. Naive Bayes: a simple probabilistic technique, based on Bayes theorem with strong independence assumptions. It gives a basic overview of how the model performs.

II. Support Vector Machine: a supervised learning technique, effective in high dimensional spaces. It learns a maximum-margin that separates the classes.

III. XGBoost: an ensemble technique, that uses gradient boosting, e. g. learning the residuals of previous models and adds new models. It uses the gradient descend algorithm to minimize the loss of its sequentially added models.

IV. LightGBM: an ensemble technique, that uses tree based learning. In contrast to other tree based learning algorithm it grows the trees vertically, e.g. by its leaves.

V. Random Forest: a bagging technique, that randomly creates a number of uncorrelated decision trees. Finally, the prediction is made with a majority vote of these trees.

   Each of them requires a individual set of parameters.

Model Selection

Hyperparameter Optimization showed that lightGBM is the best performing model for our data set. In classification lightGBM has the lowest MAE. The hyperparameter optimization chose to use the optional transformer *TimeOutlierRemover()* to exclude all trials before 2005. and *scaler function* to scale it with MinMax method.

Results

The hyperopt chose the boosting type ”goss” as the best value. We got the best MAE 15.92 in the validation set was from the modified regression model and the best value we got on test set was MAE 9.63. All chosen parameters can be found in Table 6 in the appendix.

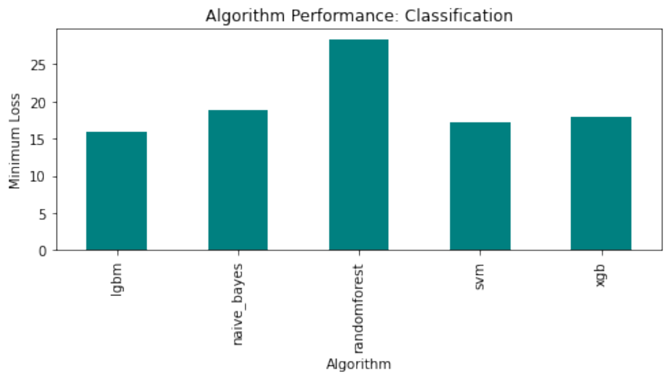


Figure 14: Algorithm Performance in Classification

4 Conclusion

4.1 Interpretation of Results

The inherited function of the lightGBM model to calculate the most important features used for prediction, confirms our assumption that the enrollment count is the most relevant feature to the enrollment duration. Figure 17 shows the 40 most important features identified according to the LGBM model.

This also indicates, that the StartYear has a high influence on the prediction. We did expect this result, as the time series split emphasises this feature. The transformer *TimeOutlierRemover()*, that was applied on the best model removes all trials before 2005. Further, we removed all trials after 2018 to avoid only having very short trials in our test set for these years. Our final training set for the best model then contained trials that started in 2005 until 2016. The test set ranges from 2016 to 2017, as it is visualized in Figure 15. Nevertheless, this could also lead to a biased prediction, as the study records in the training set have a median duration of 29 months, whereas the median duration of the test set is much smaller at 19 months.

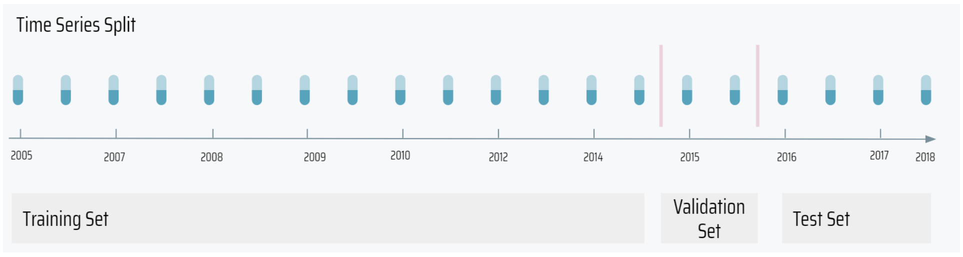
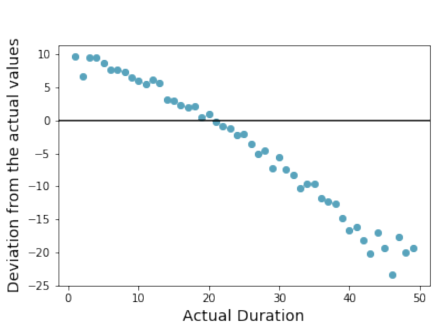
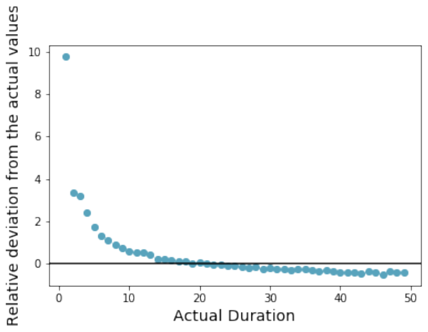


Figure 15: Visualization of the Time Series Split

Figure 16 shows the average deviation of the models prediction to the actual values, ordered by the duration of the studies. One can see in plot 16a, that although we reach an MAE of roughly 6, the accuracy of the prediction highly varies. In studies with an enrollment duration close to the mean & median of our target variable, the deviation is rather small and therefore the prediction very accurate. However, in very short or very long studies, our model on average predicts values that are much higher or much smaller than the actual values. When calculating the relative deviation from the actual values, as shown in the plot 16b, it becomes apparent that the deviation is particularly serious when predicting short studies, since an absolute deviation of 10 has a greater influence on the result here than in longer studies.
To overcome this problem and prevent our model to overfit on the medium or average sized studies, while producing large errors on very short or large studies,



(a) Average deviation from the predicted to the actual values



(b) Average relative deviation from the predicted to the actual values

Figure 16: Deviations from the predicted to the target values

consideration should be given to using the root mean square error (RMSE) instead of the MAE as an evaluation metric for model performance, as this metric gives more weight to large errors.

4.2 Concluding Remarks

In this project we scraped clinical trials data and stored it in a mongodb and gathered additional data to enrich our main data set. We created a pipeline to preprocess the individual feature categories in various ways to utilize them in the most effective way. We created an optimization framework to select the best additional preprocessing steps and the most accurate model with its optimal parameters.

We faced challenges due to the high dimensionality of the data set. We had to make a preliminary selection of features, even though many were hard to understand. Most of the features occur in lists, i.e. they have at least one, often more than 10 values. We solved this for most of them with multiple one hot encoding. Some fields have more than 10000 distinct values, which would blow up the dimensions of our final dataframe. We also tried the approach of target encoding which performed very well. The model using the target encoding delivered ultimately our best prediction. Additionally we utilized these fields, by applying a the *Distinct-Counter()* transformer, that counts the number of unique values in a list field. This way, new valuable information, such as the number of different countries and the number of arm groups. Both of them belong to the 10 most important features in our final prediction.

We came up with new ideas to enrich our data. Matching the features on the location features was the most used approach. Since all of the location related fields are list fields, we had to match the additional information with a list of countries, which created another list. To make this readable for the regression algorithms, we took the minimum, maximum and average value of each added feature. You can see in 17 in the appendix, that average and maximal features, such as average city population or maximum immigrants were important for the prediction. We also added information on the facility level. The average worldrank, assigned based on the extend of a university or hospitals research activities, is one of the 10 most important factors in our best model.

During the project we had some ideas that were quite complex to implement and dedicated a lot of work to it. For example the population density, for which we first had to retrieve the coordinates from each city with the Google Places API and then match it with the coordinates in the SEDAC population density data. On the other hand we also tried naive approached, which were more simple to implement. For example, we simply counted the number of words in the free text field eligibility criteria. We assumed, that a trials with many words, will have many restrictions on the enrollment eligibility. Another naive approach was using the MeshID, an ID for medical conditions, as a numeric value. In many cases, ID's follow a naming convention and therefore have some meaning behind the incomprehensible numbers. Finally, these two simple approaches became two of the most important features in our prediction.

4.3 Future Steps

We came up with ideas how the prediction accuracy could be further improved, that exceeded the scope of this project. One problem we faced in this project is, that we do not have differentiated information about the enrollment- and study duration. Therefore we can only roughly estimate the duration of enrollment only. In literature, an enrollment time of 30% of the whole study duration is used as a reference value. One way to solve this problem, could be to use natural language processing methods to extract the actual enrollment duration from different free text fields. Since these are written not uniformly there is neither a guiding structure to work with nor a guarantee that the enrollment duration is mentioned at all. Furthermore, the number of screened patients and eligible patients could be extracted in this way.

Another approach to improve our model is to keep enriching the clinical trials data. The possible data sets, one could use to create new features are infinite. We think the most useful information to add is more detailed information about the facilities. Currently only the feature "FacilityWorldrank" refers to the LocationFacility. The problem here is that the universities, hospitals, research centers and companies that conduct the studies enter their names in a free text field. It is already hard to match the information we already gathered on hospitals and universities with their names in the trial. E.g. The official name ”NewYork-Presbyterian / Weill Cornell Medical Center” is entered as ”Department of Ophthalmology, New York Hospital-Cornell Medical Center”. More complex NLP methods could be a good approach to solve this. Another aspect we did not include yet, is additional information about the disease being treated in the clinical trial. Matching information like the mortality rate, the percentage of affected people in a population or wheter its a acute or chronic condition. We did not pursue this idea, because it was difficult to find comprehensive data and also to match it with the condition names in the clinical trials.

The clinical trials database is maintained by the US National Institute of Health, the percentage of studies conducted by the US is predominant. 57% of the completed trials have a location in the US, while only 12% have a location in Germany. Most countries have local databases for clinical trials. It would be interesting to access their data as well, even though it would be difficult to merge them, since they do not have the exact same features. There is, for example a chinese database with 6900 completed studies[15] Finally, the search space of our hyperparameter optimization could be extended to other models. We focused on ensemble methods for this problem. Using a Neural Network is also a promising approach for this data set.

4.4 Lessons Learned

The things we learned during this project are important lessons for our further careers and are valuable experiences for us. Building up knowledge about clinical trials showed us that you really have to know the project domain to understand the problem and therefore find the best approach to solve it. In our case, accumulating knowledge about study types, eligibility criteria, phases of a trial and all of the possible factors influencing the enrollment duration was very important for us. Finding out which additional data could be useful and how to retrieve it from various API’s and crawling data from websites without API was a new experience for us as well. Retrieving the trials data and finding a convenient way to store it can be a challenge if you never faced such tasks before.

When we decided for the MongoDB we needed to get familiar with this storage technology. Overall, we build up many new skills in the field of data management.Another topic which was completely new to us, but will be an important tool in our data science skill set is the usage of a pipeline. We learned, that even though it takes some time to set it up and figure out how to use and write the transformers,

[15] www.chictr.org.cn

it is a very practical way to approach a complex data mining project like this. We are now able to assess future data science projects in a more professional and systematic way.

Another skill we gained is not related to the technologies or the framework, but is important to us on the interpersonal level. We learned how to run a whole project over 6 month without ever meeting each other in real life. Being limited by the distance regulations due to covid-19, we neither met our team mates nor our project supervisors. We learned how to adapt to these special circumstances by managing this project only in online meetings, chats and Skype calls. Finally, we also had to learn that not everything is predictable. Sometimes we have to accept that no matter how good your preprocessing or how time consuming your hyper parameter optimization is, you cannot make your predictions any more precise due to the inherent randomness of the world.

5 Appendix

5.1 Basic Statistics of all attributes Trial data

Total number of records: 30.973

Attribute Name	Missing in %	Data Type	Distinct Values	Example	Minvalue	Maxvalue	Average/ Maj. class
EnrollmentDuration	0	INT	241	35	1	306	38.55
NCTId	0	String(FT)	37604	NCT00128167	-	-	-
OrgFullName	0	String(CAT)	4536	Vicus Therap.	-	-	-
OrgClass	0	String(CAT)	8	NIH	-	-	INDUSTRY
BriefTitle	0	String(FT)	37313	-	-	-	-
OfficialTitle	1,82	String(FT)	36601	-	-	-	
BriefSummary	0	String(FT)	-	-	-	-	-
Phase	0	List(CAT)	6	-	Phase 1	Phase 4	Phase 2
StartDate	0	String	360	-	April 1991	Sept. 2019	January 2008
Condition	0	List	12562	Meningioma	-	-	Breast Cancer
ConditionAncestorTerm	8,74	List	1144	Brain Diseases	-	-	Neoplasms
ConditionBrowse-BranchAbbrev	4,68	List(CAT)	27	BC20	-		- All
ConditionBrowse-LeafRelevance	4,68	List(CAT)	2	-	-		- High
ConditionMeshId	7,34	List(NUM)	2421	D000001536	-	-	D000001943
LeadSponsorName	0	List(FT)	5444	Clinique Juge	-	-	Pfizer
LeadSponsorClass	0	List(CAT)	8	NIH	-	-	INDUSTRY
CollaboratorName	68,9	List(FT)	4716	AlzChem	-	-	Nat. Cancer Ins.
CollaboratorClass	68,9	List(CAT)	9	NIH	-	-	OTHER
EligibilityCriteria	0,01	String(FT)	-	-	-	-	-
EnrollmentCount	0	INT	2005	-	1	77535	246.395
EnrollmentType	5,85	String(CAT)	3	None	-	-	Actual
HealthyVolunteers	0,46	String(CAT)	3	Accepts HV	-	-	No
Gender	0,07	String(CAT)	4	Female	-	-	All
StdAge	0	List(CAT)	3	Child	-	-	Adult
MinimumAge	4,52	String	118	40 Years	-	-	18 Years
MaximumAge	51,69	String	215	6 Years	-	-	65 Years
LocationFacility	17,39	List(FT)	160953	Louis	-	-	Research Site
LocationCity	9,06	List(CAT)	25989	Irvine	-	-	Houston
LocationState	32,68	List(CAT)	4200	Florida	-	-	California
LocationCountry	9,06	List(CAT)	166	Canada	-	-	United States

InterventionType	0	String(CAT)	11	Device	-	-	Drug
InterventionName	0	String(FT)	35194	Atropine	-	-	Placebo
IsFDARegulatedDevice	88,17	String(CAT)	3	Yes	-	-	No
IsFDARegulatedDrug	88,11	String(CAT)	3	None	-	-	Yes
EventsTimeFrame	73,01	List(FT)	6351	3 years	-	-	1 year
FlowDrop-WithdrawType	71,08	List(FT)	7183	-	-	-	Withdrawal by Subject
FlowGroupDescription	60,64	List(FT)	28258	-	-	-	Placebo
FlowGroupTitle	60,23	String(FT)	22196	Placebo First	-	-	Placebo
FlowMilestoneType	60,23	List(FT)	2159	STARTED	-	-	COMPLETED
FlowPeriodTitle	60,23	List(FT)	3399	-	-	-	Overall Study
FlowRecruitmentDetails	81,44	List(FT)	5579	-	-	-	-
ArmGroupDescription	29,36	List(FT)	45098	-	-	-	Placebo
ArmGroup-InterventionName	13,29	List(FT)	27838	-	-	-	Placebo
ArmGroupLabel	13,29	List(FT)	36339	Axitinib	-	-	Placebo
ArmGroupType	13,29	List(FT)	7	Other	-	-	Experimental
BaselineCategoryTitle	60,72	List(FT)	3130	-	-	-	Male
BaselineGroup-Description	60,23	List(FT)	25863	-	-	-	Total of all reporting groups
BaselineGroupTitle	60,23	List(FT)	19828	Melatonin Group	-	-	Total
BaselineMeasure-DispersionType	64,76	List(CAT)	4	Full Range	-	-	Standard Deviation
BaselineMeasureTitle	60,23	List(FT)	9061	Age	-	-	Sex: Female, Male
BaselineMeasure-UnitOfMeasure	60,25	List(FT)	1482	Litters	-	-	Participants
DesignAllocation	21,05	List(CAT)	3	-	-	-	Randomized
DesignInterventionModel	2,35	List(CAT)	6	-	-	-	Parallel Assignment
DesignPrimaryPurpose	1,12	List(CAT)	11	-	-	-	Treatment
Keyword	31,04	List(FT)	38240	-	-	-	HIV

5.2 Basic Statistics of all attributes Country data

Total number of records: 235

Attribute Name	Missing in %	Data Type	Distinct Values	Example	Minvalue	Maxvalue	Average/ Maj. class
urbanPopulation	0.07	INT	81	24	0	100	59.02
countryName	235	String	235	Angola	-	-	-
population	0.004	Float	234	3480.0	801.0	1439323776.0	33315152.88
density	0.01	Float	163	80.0	2.0	26337.0	477.24
sizeInKm2	0.01	Float	224	9.0	1.0	16376870.0	560389.86
worldshare	0.01	Float	75	1.62	0.0	18.0	
lifeExpectancy	0.14	Float	33	74.0	52.0	85.0	72.12
GDP	0.12	Float	208	-	42587778.0	20544343456936.0	407457381071.36
migrantsNet	0.16	Float	158	-79.0	-653249.0	954806.0	5.17
unemploymentRate	0.23	Float	23	13.0	0.0	28.0	6.58
hospitalBed	0.17	INT	14	6	0.0	18.0	2.66
healthExpenditure	0.20	Float	16	8.0	1.0	17.0	6.11
fertilityRate	0.16	Float	51	4.1	1.1	7.0	2.69
medianAge	0.16	Float	55	5.4	1.2	44.0	3.68

5.3 Best Hyperparameters for Regression

Hyperparameter	Best Value
boosting_type	dart
colsample_bytree	0.84
eval_metric	MAE
learning_rate	0.11838544576670731
max_depth	31
max_bin	29
min_child_samples	20
min_child_weight	0.8559949533710749
min_split_gain	6.344223824193252
n_estimators	143
num_leaves	34
objective	tweedie
other_rate	0.2895773771481206
reg_alpha	3.839454095052429
reg_lambda	3.4431327423619122
subsample	0.6000000000000001
subsample_for_bin	16667
top_rate	0.41901292453458405

Table 5: The Best Model Hyperparameters

5.4 Best Hyperparameters for Classification

Hyperparameter	Best Value
boosting_type	goss
colsample_bytree	0.55
eval_metric	MAE
learning_rate	0.13510592140551062
max_depth	86
max_bin=	24
min_child_samples	20
min_child_weight	2.779265740273007
min_split_gain	6.216084314997958
n_estimators	105
num_leaves	10
objective	huber
other_rate	0.4267690296035428
reg_alpha	0.3757383393588283
reg_lambda	4.577713645388543
subsample	0.75
subsample_for_bin	19186
top_rate	0.15498341452300046

Table 6: The Best Model Hyperparameters for Classification

5.5 Most important features

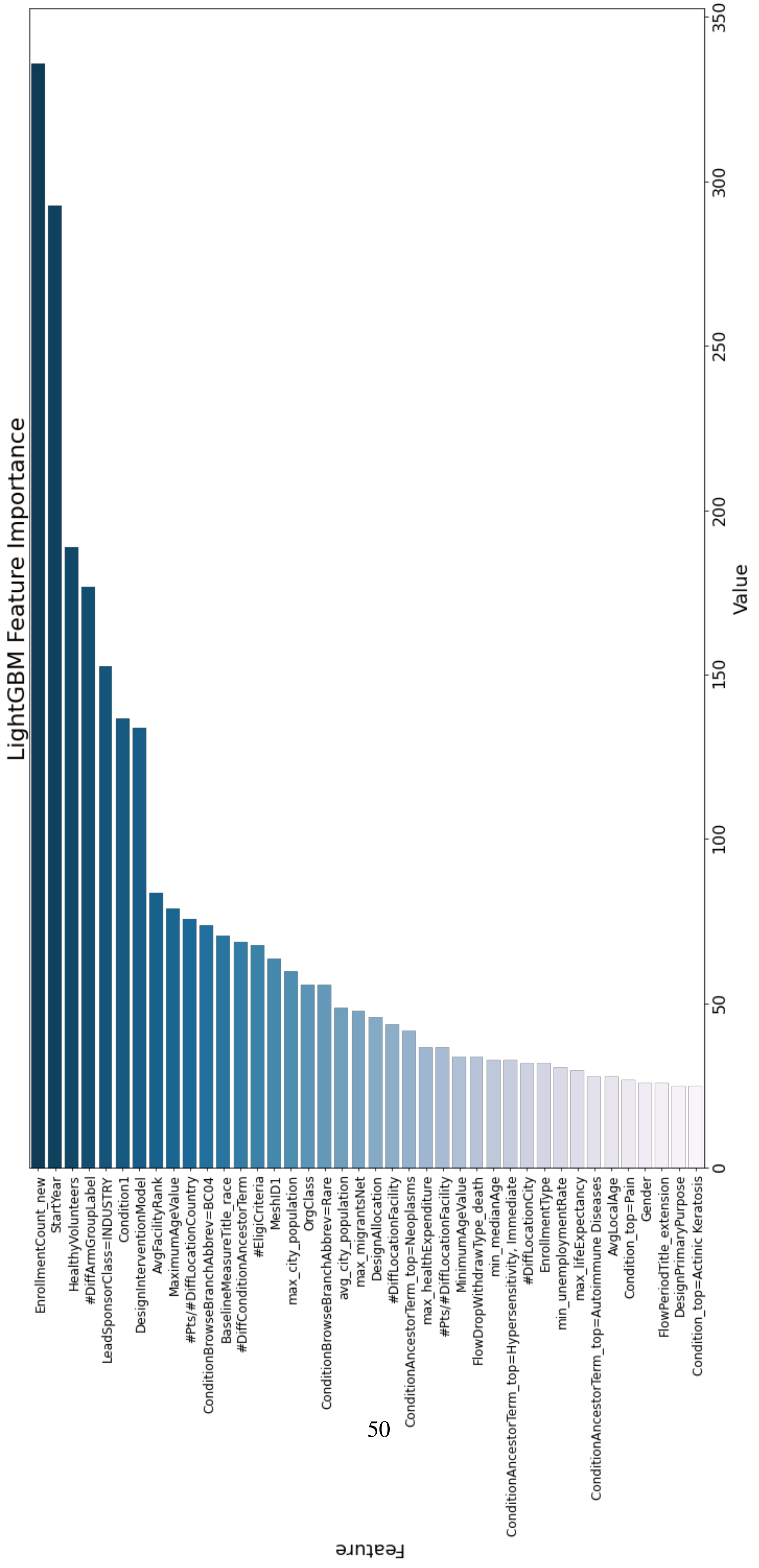


Figure 17: The most important Features

References

[1] Socioeconomic Data and Applications Center. Population density v4.11. https://sedac.ciesin.columbia.edu/data/set/gpw-v4-population-density-rev11. Accessed: 20.05.2020.

[2] H. Eichler and F. Sweeney. The evolution of clinical trials: Can we address the challenges of the future? 15(S1):27–32, 2018.

[3] Lawrence M. Friedman, Curt Furberg, and David L. DeMets. *Fundamentals of clinical trials*. Springer New York, 5. ed edition. OCLC: 935903160.

[4] David D. Cox James Bergstra, Dan Yamins. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. *PROC. OF THE 12th PYTHON IN SCIENCE CONF. (SCIPY 2013)*.

[5] Andrew W. Lo, Kien Wei Siah, and Chi Heem Wong. Machine learning with statistical imputation for predicting drug approvals. *Harvard Data Science Review*, 1(1), 7 2019. https://hdsr.mitpress.mit.edu/pub/ct67j043.