Logic Models to Classify Cancer Types based on Tumor DNA Samples

Nguyen Khoa Tran

A thesis presented for the degree of
Master of Science

Algorithmic Bioinformatics
Heinrich Heine University Düsseldorf
Germany
5th January, 2021
Acknowledgments

I would like to express my sincerest gratitude to my first assessor Prof. Dr. Gunnar Klau for proposing the intriguing topic, providing guidance, and always being understanding in more than one way. Next, I would like to thank Prof. Dr. Tobias Marschall for being my second assessor. Furthermore, I owe special thanks to my colleagues Laura Kühle, Eline van Mantgem, Sven Schrinner, and Philipp Spohr, as well as our administrator Max Ried for insightful discussion and valuable help.

Computational infrastructure and support were provided by the Center for Information and Media Technology at Heinrich Heine University Düsseldorf.

Finally, I would like to thank my family, boyfriend, friends, all living and deceased beings, or to sum it up, Mother Nature. Thank you for making it possible for me to live every day to its fullest!
Abstract

Correctly predicting the cancer type of a tumor DNA sample can aid in cancer research, diagnostics, and therapy. In order to create explainable logic models with high credibility and interpretability that predict these cancer types, we introduce “LOCATOR”, which uses boolean logical formulas in disjunctive normal form (DNF) for predictions, with each literal standing for the presence or absence of a specific mutation. One DNF formula is constructed for each cancer type, and if a DNF formula is true for a tumor DNA sample, the sample is classified as the corresponding cancer type. LOCATOR uses recursive feature elimination for linear support-vector machines (SVM-RFE) to reduce the high dimensionality of cancer data, and integer linear programs (ILPs) to find the best DNF formulas.

In addition, we try to predict cancer types by finding a set of potential driver mutations for each cancer type, while considering the hypothesis that driver mutations tend to be mutually exclusive. Therefore, we introduce “CATMEME”, which also uses SVM-RFE on cancer data and ILPs for finding these sets of potential cancer-type specific, mutually exclusive driver mutations.

We used cancer data from the online genome database cBioPortal. The SVM-RFE we use reduced the dimensionality of the cancer data from 13150 to 250.

We evaluated CATMEME based on the Cancer Gene Census (CGC) from the Catalogue Of Somatic Mutations In Cancer, which lists known driver mutations for several cancer types. Among the eight tested CATMEME variants, the best one did not find many known driver mutations that are listed in the CGC, and we deem the other potential ones found improbable to be actual cancer drivers. Accordingly, classification with CATMEME does not yield good results either.

We evaluated LOCATOR based on accuracy, precision, and recall using stratified 4-fold cross-validation. Among the 16 tested LOCATOR variants, the best one in terms of accuracy and recall achieves an accuracy of 71.7 % ± 0.8, an average precision of 77.7 % ± 0.9, and an average recall of 65.6 % ± 1.3, whereas the best one in terms of precision achieves an accuracy of 62.8 % ± 1.0, an average precision of 84.6 % ± 2.8, and an average recall of 56.1 % ± 1.5. Besides, we considered the CGC for LOCATOR as well. LOCATOR finds more known driver mutations than CATMEME, and we believe that the other mutations found, but not listed in the CGC, might be cancer-related as well.

As a conclusion, predicting cancer types using boolean logical formulas can yield explainable models with high credibility and interpretability that have an acceptable accuracy, while the hypothesis of a set of mutually exclusive driver mutations for each cancer type is not supported by our results.
Contents

List of Tables \textit{v}

List of Figures \textit{vi}

List of Abbreviations \textit{vii}

1 Introduction 1

2 Preliminaries 3
  2.1 DNF Formula 3
  2.2 Support-Vector Machine 3
  2.3 Recursive Feature Elimination 4
  2.4 $k$-Fold Cross-Validation 4
  2.5 Integer Linear Programming 5
  2.6 Evaluation Metrics 5

3 Methods 6
  3.1 Data and Preprocessing 6
  3.2 LOCATOR 7
    3.2.1 Constructing Logic Models 7
    3.2.2 Classification Options 11
  3.3 CATMEME 14
    3.3.1 Establishing Sets of Class-Specific, Mutually Exclusive Features 14
    3.3.2 Detailed ILP Formulations 17

4 Results 18
  4.1 LOCATOR 18
    4.1.1 Accuracy, Precision, and Recall 19
    4.1.2 Limited Number of Literals and Minterms 23
  4.2 CATMEME 26

5 Discussion 28
  5.1 Data and Preprocessing 28
  5.2 LOCATOR 28
    5.2.1 Modifications 28
    5.2.2 ILP Formulations 29
    5.2.3 Classification Options 30
    5.2.4 Explainability 31
    5.2.5 Credibility 31
    5.2.6 Interpretability 32
  5.3 CATMEME 34

6 Conclusions 35

7 References 36
## List of Tables

1. Extract from the data provided by Soh et al. Note that we label the classes from 0 to 27 instead of 1 to 28. ......................................................... 6
2. Overview of modification effects for LOCATOR. Note that modification is abbreviated to “mod”. ................................................................. 9
3. Overview of ILP formulations for LOCATOR and their effects. Note that ILP formulation is abbreviated to “ILP”. ................................. 11
4. Overview of ILP formulations for CATMEME and their penalties depending on which conditions hold true for \( y_n \) and \( F \), where \( F = \sum_{m=1}^{M} X_{nm} \cdot s_m \). Each entry is a penalty value. Note that ILP formulations 1 to 4 are identical to ILP formulations 5 to 8 if we disregard the fourth column \( y_n = 0, F \geq 2 \). .......................... 16
5. Accuracy, average precision, and average recall of LOCATOR 3\(_1\), LOCATOR 4\(_{5-2}\), SVM with \( C = 0.02 \), and Soh et al.’s SVM-RFE. The values after ± denote the standard deviations. Note that for Soh et al.’s SVM-RFE, the standard deviations are not fully comparable due to a different experimental set-up as described in Section 4.1. Both \( L \) and \( P \) were not limited for LOCATOR 3\(_1\) and LOCATOR 4\(_{5-2}\). The results were obtained via stratified 4-fold cross-validation, except for Soh et al.’s SVM-RFE. ........................................ 21
6. Accuracy for all datapoints with ambiguous class of each classification option with LOCATOR 3 and 4. The values after ± denote the standard deviations. The last row shows the expected accuracy when classifying a datapoint with ambiguous class randomly as one of the classes that come into question. Note that for simplicity, we assumed that the correct class is always among those classes that come into question. ........................................ 22
7. All results of CATMEME. The first column lists all classes for which the ILP formulations listed in the second column found a set of potential class-specific, mutually exclusive features. The third column lists the objective values of the corresponding ILP formulations per class. The fourth column lists the number of datapoints in the corresponding class. The fifth column lists the number of known driver mutations per class. For each class, the sixth column lists the number of potential class-specific, mutually exclusive features in a set. The last column lists the overlap between a set of class-specific, mutually exclusive features and the class-specific known driver mutations. The results were obtained by applying each CATMEME variant on the whole dataset. .......................... 27
8. Average number of minterms and literals per minterm for each LOCATOR variant with unlimited \( L \) and \( P \). Using no modification is denoted as “-”. Note that modification is abbreviated to “mod”. The results were obtained via stratified 4-fold cross-validation. ........................................ 30
Overlapping mutations between LOCATOR 4 and the CGC. The second column lists the number of class-specific known driver mutations in the CGC per class after filtering out those mutations that do not appear in the dataset. The third column lists the number of positive literals per class. The last column lists overlaps between positive literals, which correspond to mutations, and the class-specific known driver mutations per class. The results were obtained by applying LOCATOR 4 on the whole dataset with unlimited $L$ and $P$.

List of Figures

1. Example of linear SVM. There are 2-dimensional datapoints of three classes (red, green, blue), which are separated by three hyperplanes for each class, which are the lines colored in the respective color. In particular, these hyperplanes are linear because we have a linear SVM.

2. LOCATOR workflow. Note that only the core components are displayed.

3. CATMEME workflow. Note that only the core components are displayed.

4. (a) Accuracy, (b) average precision, and (c) average recall of all 16 LOCATOR variants per classification option. The $y$-axis shows the evaluation metric. The $x$-axis shows the 15 classification options from Section 3.2.2. Note that modification is abbreviated to “mod”. Both $L$ and $P$ were not limited. The results were obtained via stratified 4-fold cross-validation.

5. Precision and recall of (a) SVM with $C = 0.02$, (b) Soh et al.’s SVM-RFE (adapted from [1]), (c) LOCATOR 3$^1$, and (d) LOCATOR 4$^{5,2}$. The $y$-axis shows classes, i.e., cancer types. The blue bars show precision, the orange bars show recall. Both $L$ and $P$ were not limited for (c) and (d). The results were obtained via stratified 4-fold cross-validation, except for (b) as they are adapted.

6. (a) Accuracy, (b) average precision, and (c) average recall of all 16 LOCATOR variants per classification option. The $y$-axis shows the evaluation metric. The $x$-axis shows the 15 classification options from Section 3.2.2. Note that modification is abbreviated to “mod”. Both $L$ and $P$ were limited to 10. The results were obtained via stratified 4-fold cross-validation.

7. (a) Accuracy, (b) average precision, and (c) average recall of all 16 LOCATOR variants per classification option. The $y$-axis shows the evaluation metric. The $x$-axis shows the 15 classification options from Section 3.2.2. Note that modification is abbreviated to “mod”. $L$ was limited to 10 and $P$ was limited to 5. The results were obtained via stratified 4-fold cross-validation.

8. Precision and recall of CATMEME 1 with classification option 1. The $y$-axis shows classes, i.e., cancer types. The blue bars show precision, the orange bars show recall. The results were obtained by applying CATMEME 1 on the whole dataset.
# List of Abbreviations

## General Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CATMEME</strong></td>
<td>Cancer-type specific, mutually exclusive mutations establisher</td>
</tr>
<tr>
<td><strong>CGC</strong></td>
<td>Cancer Gene Census</td>
</tr>
<tr>
<td><strong>CNA</strong></td>
<td>Copy-number alteration</td>
</tr>
<tr>
<td><strong>CPU</strong></td>
<td>Central processing unit</td>
</tr>
<tr>
<td><strong>DNA</strong></td>
<td>Deoxyribonucleic acid</td>
</tr>
<tr>
<td><strong>DNF</strong></td>
<td>Disjunctive normal form</td>
</tr>
<tr>
<td><strong>FN</strong></td>
<td>False negative</td>
</tr>
<tr>
<td><strong>FP</strong></td>
<td>False positive</td>
</tr>
<tr>
<td><strong>GB</strong></td>
<td>Gigabyte</td>
</tr>
<tr>
<td><strong>Gbit</strong></td>
<td>Gigabit</td>
</tr>
<tr>
<td><strong>GPFS</strong></td>
<td>General Parallel File System</td>
</tr>
<tr>
<td><strong>ILP</strong></td>
<td>Integer linear program</td>
</tr>
<tr>
<td><strong>LOCATOR</strong></td>
<td>Logical cancer-type predictor</td>
</tr>
<tr>
<td><strong>PB</strong></td>
<td>Petabyte</td>
</tr>
<tr>
<td><strong>RAM</strong></td>
<td>Random-access memory</td>
</tr>
<tr>
<td><strong>RFE</strong></td>
<td>Recursive feature elimination</td>
</tr>
<tr>
<td><strong>SPM</strong></td>
<td>Somatic point mutation</td>
</tr>
<tr>
<td><strong>SVM</strong></td>
<td>Support-vector machine</td>
</tr>
<tr>
<td><strong>SVM-RFE</strong></td>
<td>Recursive feature elimination for linear support-vector machines</td>
</tr>
<tr>
<td><strong>TN</strong></td>
<td>True negative</td>
</tr>
<tr>
<td><strong>TP</strong></td>
<td>True positive</td>
</tr>
<tr>
<td>Cancer Type Abbreviations</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>BLCA</td>
<td>Bladder urothelial carcinoma</td>
</tr>
<tr>
<td>BRCA</td>
<td>Breast invasive carcinoma</td>
</tr>
<tr>
<td>COADREAD</td>
<td>Colorectal adenocarcinoma</td>
</tr>
<tr>
<td>GBM</td>
<td>Glioblastoma</td>
</tr>
<tr>
<td>HNSC</td>
<td>Head and neck squamous cell carcinoma</td>
</tr>
<tr>
<td>KIRC</td>
<td>Kidney renal clear cell carcinoma</td>
</tr>
<tr>
<td>LAML</td>
<td>Acute myeloid leukaemia</td>
</tr>
<tr>
<td>LUAD</td>
<td>Lung adenocarcinoma</td>
</tr>
<tr>
<td>LUSC</td>
<td>Lung squamous cell carcinoma</td>
</tr>
<tr>
<td>OV</td>
<td>Ovarian serous cystadenocarcinoma</td>
</tr>
<tr>
<td>UCEC</td>
<td>Uterine corpus endometrial carcinoma</td>
</tr>
<tr>
<td>AdCC</td>
<td>Adenoid cystic carcinoma</td>
</tr>
<tr>
<td>LGG</td>
<td>Brain lower grade glioma</td>
</tr>
<tr>
<td>CESC</td>
<td>Cervical squamous cell carcinoma and endocervical adenocarcinoma</td>
</tr>
<tr>
<td>KIRP</td>
<td>Kidney renal papillary cell carcinoma</td>
</tr>
<tr>
<td>LIHC</td>
<td>Liver hepatocellular carcinoma</td>
</tr>
<tr>
<td>PAAD</td>
<td>Pancreatic adenocarcinoma</td>
</tr>
<tr>
<td>PRAD</td>
<td>Prostate adenocarcinoma</td>
</tr>
<tr>
<td>SKCM</td>
<td>Skin cutaneous melanoma</td>
</tr>
<tr>
<td>STAD</td>
<td>Stomach adenocarcinoma</td>
</tr>
<tr>
<td>PTC</td>
<td>Papillary thyroid carcinoma</td>
</tr>
<tr>
<td>ACC</td>
<td>Adrenocortical carcinoma</td>
</tr>
<tr>
<td>KICH</td>
<td>Kidney chromophobe</td>
</tr>
<tr>
<td>PCPG</td>
<td>Pheochromocytoma and paraganglioma</td>
</tr>
<tr>
<td>SARC</td>
<td>Sarcoma</td>
</tr>
<tr>
<td>TGCT</td>
<td>Testicular germ cell cancer</td>
</tr>
<tr>
<td>UCS</td>
<td>Uterine carcinosarcoma</td>
</tr>
<tr>
<td>UVM</td>
<td>Uveal melanoma</td>
</tr>
</tbody>
</table>
1 Introduction

Characterizing tumor DNA from cancer cells poses a big challenge for cancer research, diagnostics, and therapy. Establishing the primary site (body area where the cancer started) of a tumor DNA sample can facilitate this challenge. However, a medical examination is not always able to establish a primary site because somatic mutations in tumor DNA are very complex such that the cancer data is highly dimensional.

Soh et al. [1] evaluated the potential and limitations of using information on somatic mutations in tumor DNA samples to predict cancer types, which are named after their primary site, by using machine learning approaches. Among the approaches they considered, support-vector machine models perform best and make accurate predictions. However, the underlying functions of these models are hard to explain and interpret, and therefore unfit for gaining further knowledge about cancer. Thus, we pose the following research question:

*How can we construct explainable models with high credibility and interpretability to aid in cancer research, diagnostics, and therapy?*

In order to construct such models, we present a new machine learning approach called “LOgical CAncer-type predicTOR” (LOCATOR), which uses boolean logical formulas to locate the primary site. These formulas consist of literals, each standing for a somatic mutation in a certain gene (DNA segment). LOCATOR combines support-vector machines for reducing the high dimensionality of tumor DNA data and mathematical optimization for finding suitable logical formulas.

Besides, we try to establish the cancer type of a tumor DNA sample by distinguishing driver mutations, which are associated with the start of cancer, from passenger mutations, which are not. A widely accepted hypothesis is that driver mutations with similar cancer-driving effects tend to be mutually exclusive [2, 3]. We speak of mutual exclusivity for a set of mutations if the mutations in that set co-occur considerably less often than expected by chance [4]. Our hypothesis is that each cancer type might have different sets of mutually exclusive driver mutations, which can be used for predicting cancer types. Hence, we present another new machine learning approach called “CAncer-Type specific, Mutually Exclusive Mutations Establisher” (CATMEME) to establish a set of potential mutually exclusive mutations for each cancer type. CATMEME also aims at constructing explainable models with high credibility and interpretability by using mathematical optimization.

In Section 2, we give an overview of the theoretical concepts that are needed for understanding this thesis.

Section 3 describes the data we use for LOCATOR and CATMEME, which includes data from the online genome database cBioPortal [5] and the Cancer Gene Census (CGC) [6, 7]. Furthermore, we explain several variants of LOCATOR and CATMEME.

Our results are presented in Section 4, where we evaluate all LOCATOR variants based on accuracy, precision, and recall using stratified 4-fold cross-validation for the cancer data from
the cBioPortal. Furthermore, we compare the performances of LOCATOR models and support-vector machine models. The CATMEME variants are evaluated based on how many known driver mutations from the CGC are found using the whole cancer data from the cBioPortal.

A critical reflection on our research is given in Section 5 and deals with all LOCATOR and CATMEME variants. Moreover, we discuss the explainability, credibility, and interpretability of LOCATOR, while considering the CGC.

The last section summarizes our findings and lists possible future work.
2 Preliminaries

This section gives a short overview of the theoretical concepts that are needed for a thorough understanding of this thesis. The biological terms are abstracted as follows: a tumor DNA sample becomes a datapoint, a somatic mutation in a gene becomes a feature, and a cancer type becomes a class. That is, each biological term and its abstraction can be used interchangeably.

2.1 DNF Formula

In this thesis, we use boolean logical formulas for classification, which consist of literals: a positive literal is a variable representing a statement, whereas a negative literal is the negation of a positive literal and identifiable by a preceding logical ¬-gate. A minterm (also called product) connects literals with ∧-gates and is true if all literals are true. A boolean logical formula in disjunctive normal form (DNF) connects minterms with ∨-gates and is true if at least one minterm is true. Note that all boolean logical formulas can be rewritten as DNF formulas. An example for a DNF formula Φ that takes as input three literals a, b, and c is Φ(a, b, c) = (a ∧ b) ∨ (a ∧ ¬c).

2.2 Support-Vector Machine

Support-vector machine (SVM) is a technique from the field of machine learning. For classification problems, machine learning approaches construct/train a mathematical model using datapoints with known class label such that these models can predict the class labels of datapoints with unknown class label. For a datapoint, each feature represents a dimension in an n-dimensional vector space. An SVM model is trained to maximize correct classification of datapoints according to their class labels by fitting hyperplanes into this space and thus separating the datapoints. A hyperplane is an (n − 1)-dimensional subspace described by one support vector and n − 1 linearly independent direction vectors, where all vectors are n-dimensional. Soh et al. [1] use the one-vs-all strategy, meaning that they fit a hyperplane for each class that separates this class from all other classes, and thus we use this strategy as well. Eventually, datapoints with unknown class label can be classified by obtaining their position in the vector space. Training an SVM model requires a regularization parameter C, where larger values for C increase the distances between hyperplanes and their closest datapoints. This increase might lead to a worse classification of known datapoints, but a better classification of unknown datapoints. Note that there is no general rule for choosing the value of C. In this thesis, we use linear SVMs, meaning that the hyperplanes are linear, see Figure 1.
2.3 Recursive Feature Elimination

Recursive feature elimination for linear SVMs (SVM-RFE) is a method for dimensionality reduction and can be divided into three steps. First, a linear SVM is trained on the dataset. Second, a freely selectable number of less significant features is eliminated, where the direction vectors describing the hyperplanes of the linear SVM are considered, because a dimension where each direction vector has an entry close to zero corresponds to a feature that is less significant for classification. Third, the first two steps may be repeated. If they are not, the features corresponding to the desired number of dimensions are kept.

2.4 k-Fold Cross-Validation

A commonly used technique for evaluating or training models is k-fold cross-validation. First, a dataset is randomly split into $k \geq 3$ folds (disjoint subsets) of roughly equal size, where stratified k-fold cross-validation additionally preserves the class proportions for each fold. One of the folds acts as the test set, while the others form the training set for training the model. Next, the trained model is evaluated on the test set with one or more evaluation metrics. This procedure is repeated $k$ times until each fold acted as the test set exactly once. For each evaluation metric, the $k$ values are finally combined into one value. In this thesis, we combine values into one value by taking their mean.
2.5 Integer Linear Programming

An integer linear program (ILP) is a mathematical program for optimization problems and is expressed as:

\[
\begin{align*}
\text{max } & \quad c^T x \\
\text{s. t. } & \quad Ax \leq b \\
& \quad x \in \mathbb{N}_0^n
\end{align*}
\]

ILPs consist of a linear objective function (1) subject to linear constraints (2), and variables that are restricted to be non-negative integers (3). The constraints define the solution space. Solving an ILP means optimizing the objective function while staying within the solution space. In order to do this, an ILP solver determines suitable values for the variables in vector \( x \). Matrix \( A \) as well as vectors \( b \) and \( c \) are the input for an ILP. Note that different expressions exist for ILPs, which can all be converted into one another.

2.6 Evaluation Metrics

We employ the following three evaluation metrics for models: accuracy, precision, and recall.

We define the accuracy of a model as follows:

\[
\text{accuracy} = \frac{\text{number of correctly classified datapoints}}{\text{total number of datapoints}}
\]

For a class \( c \), true positives (TPs) are datapoints correctly classified as class \( c \), true negatives (TNs) are datapoints correctly classified as a class other than \( c \), false positives (FPs) are datapoints wrongly classified as class \( c \), and false negatives (FNs) are datapoints wrongly classified as a class other than \( c \).

The precision of a model for class \( c \) shows how credible the predictions are and is defined as:

\[
\text{precision}_c = \frac{\text{number of TPs}}{\text{number of TPs and number of FPs}}
\]

If there are neither TPs nor FPs, i.e., if there is a division by zero, we set \( \text{precision}_c \) to 0 in this thesis.

The recall of a model for class \( c \) shows how complete the predictions are and is defined as:

\[
\text{recall}_c = \frac{\text{number of TPs}}{\text{number of TPs and number of FNs}}
\]

If there are neither TPs nor FNs, i.e., if there is a division by zero, we set \( \text{recall}_c \) to 0 in this thesis.
3 Methods

In this section, we describe the input data as well as the workflows of LOCATOR and CATMEME.

3.1 Data and Preprocessing

We use the data provided by Soh et al. [1], which was downloaded from cBioPortal [5] and includes two types of somatic mutations: somatic point mutations (SPMs) and copy-number alterations (CNAs). SPMs are substitutions of single nucleotide bases in the DNA, whereas CNAs are amplifications or deletions of DNA parts. The dataset consists of 6640 rows and 13152 columns. Each row represents a tumor DNA sample. The first column stores the tumor DNA sample names. The last column stores the class labels ranging from 1 to 28 for identifying the cancer type of a tumor DNA sample. In this thesis, we label the 28 classes from 0 to 27 instead. All remaining 13150 columns store the presence (= 1) or absence (= 0) of a somatic mutation in a gene of a tumor DNA sample. Each of those columns was named by Soh et al. as follows. If the somatic mutation in a gene was a CNA, the column was named after the gene, but extended by ".p" or ".n", depending on whether an amplification or a deletion occurred, respectively. If the somatic mutation in a gene was an SPM, the column was simply named after the gene. Note that Soh et al. removed all columns that contained only zeros, and subsumed all identical columns into one. Table 1 shows an extract from the data.

<table>
<thead>
<tr>
<th>A1CF</th>
<th>A2M</th>
<th>...</th>
<th>ZSWIM7.p</th>
<th>ZSWIM7.n</th>
<th>classLabel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>...</td>
<td>1</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td>27</td>
</tr>
</tbody>
</table>

Table 1: Extract from the data provided by Soh et al. Note that we label the classes from 0 to 27 instead of 1 to 28.

As the dataset would be too large for training LOCATOR and CATMEME models, we reduce its dimensionality with SVM-RFE. We apply the LinearSVC tool for training linear SVMs, where we use $C = 1$, in combination with the RFE tool from the Scikit-learn package [8] on our dataset. In the first SVM-RFE step, we remove up to nine features such that our number of features is divisible by ten. In each further step, we eliminate ten features until we reach zero. We keep as few features as possible such that the accuracy of LinearSVC for the dataset with those few features is not less than the accuracy of LinearSVC for the dataset with all features. These accuracy values are determined via 5-fold cross-validation. Note that all of our choices regarding the application of SVM-RFE are arbitrary.
Besides, for each of the 28 cancer types, we compiled a list of known driver mutations using the first tier of the CGC [6, 7]. This had to be done manually because the 28 cancer types are subtypes of the more general types listed in the CGC. Note that for our compiled lists, we did not consider the mutations in the second tier of the CGC since not all mutations in the second tier are evidently related to cancer.

3.2 LOCATOR

Figure 2 shows the LOCATOR workflow, which can be briefly described as follows: LOCATOR requires data where the last column stores class labels and the remaining columns are features that are either present (= 1) or absent (= 0). Next, the dimensionality of the data is reduced with SVM-RFE as described in Section 3.1. Afterwards, a DNF formula is constructed for each class, see Section 3.2.1. The idea behind constructing a DNF formula for a cancer type is that each minterm represents a driver pathway, and that the literals of each minterm reveal which mutations need to be present (positive literal) or absent (negative literal). Together, these DNF formulas form a logic model that can eventually classify datapoints with unknown class label, see Section 3.2.2.

![LOCATOR Workflow Diagram](image)

**Figure 2**: LOCATOR workflow. Note that only the core components are displayed.

3.2.1 Constructing Logic Models

In this section, we present different LOCATOR variants to construct logic models. Let \( N \) be the number of datapoints and let \( M \) be the number of features. The DNF formulas use features of the data as (positive or negative) literals, i.e., we can plug a datapoint into a DNF formula for class \( c \) to obtain true (= 1) or false (= 0). In most cases, obtaining true means that the datapoint is classified as class \( c \). However, if the class of a datapoint is ambiguous, i.e., if multiple DNF formulas are true for this datapoint, we proceed as described in Section 3.2.2.

Constructing a DNF formula for class \( c \) needs the following inputs:

1. \( L \), the maximal number of literals that a minterm is allowed to consist of,
2. \( P \), the maximal number of minterms that the DNF formula is allowed to consist of,
3. \( X \), an \( N \times M \) binary matrix where an entry \( X_{nm} \) is 1 if feature \( X_m \) is present in datapoint \( X_n \),
4. \( y \), an \( N \times 1 \) binary vector where an entry \( y_n \) is 1 if datapoint \( X_n \) belongs to class \( c \).

Note that \( y \) can be easily created with the column storing the class labels.
Constructing an optimal DNF formula for class $c$ is equivalent to the “Boolean Function Synthesis Problem”, and its associated decision problem is NP-complete [9]. Accordingly, searching for an optimal DNF formula turned out to have an infeasible runtime in practice, probably because there exist up to

$$1 + \sum_{l=1}^{L} \sum_{p=1}^{P-1} \binom{l}{p} \cdot \binom{2 \cdot M + l - 1}{l}$$

possibilities to choose from, which is already huge for very small $L$ and $P$. The first part of the formula, in which we add 1, is the possibility where the optimal DNF formula contains no literals (and consequently no minterms). The second part, $\binom{l}{p}$, describes the number of possibilities of placing up to $P - 1 \lor$-gates between up to $L \cdot P$ literals. The last part, $\binom{2 \cdot M + l - 1}{l}$, is equivalent to unordered sampling with replacement as known from combinatorics. Moreover, this last part contains $2 \cdot M$ instead of just $M$ since we have $M$ features that can be used as both positive and negative literals. Although not all of those possibilities need to be checked, we did not obtain any results after one week on our used computational infrastructure, which we describe in Section 4.

Therefore, instead of searching for the optimal DNF formula, we designed a much faster heuristic approach with four steps:

1. Construct one optimal minterm with up to $L$ literals.
2. Exclude all datapoints for which the newest (most recently constructed) minterm is true.
3. If the newest minterm contains no literals, skip this step. Else, repeat both steps up to $P$ times.
4. Connect all minterms with $\lor$-gates.

LOCATOR offers the following three modifications that exclude features in addition to excluding datapoints:

A. Exclude all features that are used as a literal in the newest minterm.

B. Exclude all features that are used as a positive literal in the newest minterm.

C. Exclude all features that are used as a positive literal in the newest minterm. Whenever a negative literal in any previous minterm is used as a positive literal in the newest minterm, add the positive literals in that previous minterm to the newest minterm as negative literals. In other words, enforce mutual exclusivity between two literals $l_i$ and $l_j$ if both were used together as $l_i$ and $\neg l_j$ in any previous minterm.

For our heuristic approach, repeatedly constructing an optimal minterm without making changes to the dataset would result in always constructing the exact same optimal minterm. To avoid this, we decided to exclude datapoints for which the newest minterm is true due to two reasons. The first reason is that the DNF formula will be true for those datapoints
regardless of all other minterms. The second reason is that we were unable to run LOCATOR on our used computational infrastructure in reasonable time when we kept all datapoints and only excluded features by using any of the modifications.

The three modifications and the possibility to limit $L$ and/or $P$ aim to counteract overfitting. Overfitting means that a model is trained so precisely on the available data such that new data cannot be reliably classified anymore.

Note that for modification C, excluding features is necessary before adding literals. Otherwise, a positive literal $l_i$ and a negative literal $\neg l_j$ of a previous minterm could both be used in the newest minterm as positive literals $l_i$ and $l_j$. If we then add literals according to modification C, the newest minterm will include $l_i$ and $\neg l_j$ and thus be false for all datapoints. Consequently, LOCATOR does not exclude any datapoint, hence this minterm is repeatedly constructed.

Besides, modification C could exhibit sets of class-specific, strictly mutually exclusive feature sets (not to be confused with sets of cancer-type specific, mutually exclusive features), which means that features within a feature set can be present simultaneously, but features in different feature sets cannot. We further discuss this topic in Section 5.2.6.

Table 2 gives a summarizing overview of our modifications.

<table>
<thead>
<tr>
<th></th>
<th>exclude negative literals</th>
<th>exclude positive literals</th>
<th>enforce mutual exclusivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>no mod</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>mod A</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>mod B</td>
<td>X</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>mod C</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 2: Overview of modification effects for LOCATOR. Note that modification is abbreviated to "mod".

Next, we explain how we construct a minterm in each step. For each class $c$, we construct it with an ILP, which is adapted from LOBICO [10]. The ILP formulation of LOBICO constructs an optimal DNF formula with up to $P$ minterms as well as up to $L$ literals per minterm, and also considers weights for each datapoint in the objective function, but since we only construct one minterm without considering weights, we set the $P$ to 1 and simplified the ILP accordingly. The following four ILP formulations for LOCATOR search for an optimal minterm $\mu$ such that the number of differences between $y$ and $\mu(X)$ is minimal, with $\mu(X)$ being the $N \times 1$ binary vector that is obtained from plugging each datapoint in $X$ into $\mu$.

Given $L$, $X$, and $y$, the ILP formulations for constructing one minterm with up to $L$ literals for a class $c$ employs three sets of binary decision variables:
\[ l_m = \begin{cases} 1 & \text{if feature } X_m \text{ is used as a positive literal in the minterm,} \\ 0 & \text{else.} \end{cases} \]

\[ l'_m = \begin{cases} 1 & \text{if feature } X_m \text{ is used as a negative literal in the minterm,} \\ 0 & \text{else.} \end{cases} \]

\[ y'_n = \begin{cases} 1 & \text{if the minterm is true for datapoint } X_n, \\ 0 & \text{else.} \end{cases} \]

1. The first ILP formulation reads as follows:

\[
\begin{align*}
\min & \quad \frac{N+1}{N} \cdot \sum_{\forall n : y_n = 0} y'_n - \sum_{\forall n : y_n = 1} y'_n \\
\text{s. t.} & \quad l_m + l'_m \leq 1 \quad m = 1, \ldots, M \\
& \quad \sum_{m=1}^M l_m + l'_m \leq L \\
& \quad \sum_{m=1}^M l_m \geq 1 \\
& \quad M \cdot y'_n \leq M - \sum_{\forall m : X_{nm} = 1} l'_m - \sum_{\forall m : X_{nm} = 0} l_m \leq y'_n + M - 1 \quad n = 1, \ldots, N \\
& \quad l_m \in \{0, 1\} \quad m = 1, \ldots, M \\
& \quad l'_m \in \{0, 1\} \quad m = 1, \ldots, M \\
& \quad y'_n \in \{0, 1\} \quad n = 1, \ldots, N 
\end{align*}
\] (4)

(5)

(6)

(7)

(8)

(9)

(10)

(11)

The objective function (4) aims at minimizing FPs while maximizing TPs, where we penalize FPs slightly more than we reward TPs by multiplying \( \frac{N+1}{N} \) to the left sum. Thereby, we ensure including as few FPs as possible in case that multiple solutions would have the same objective value without \( \frac{N+1}{N} \). Constraints (5) reduce the solution space by ensuring that no literal appears more than once per minterm because this would be counterproductive \( (l_m \land l'_m) = \text{false} \). The single constraint (6) limits the number of literals to \( L \), while the single constraint (7) ensures that at least one mutation occurs since cancer does not develop without any mutation. We abuse notation for constraints (8), which encode the \( \land \)-gate. We consider two cases for each datapoint. If no present feature is used as a negative literal and if no absent feature is used as a positive literal, then the middle part of constraints (8) is \( M - 0 - 0 = M \) such that the right part forces \( y'_n \) to be 1. Otherwise, the middle part of constraints (8) sums up to some number less than \( M \) such that the left part forces \( y'_n \) to be 0.

LOCATOR offers the following three additional ILP formulations, which have the same constraints (5) to (11), but a different objective function:
2. The second ILP formulation penalizes FPs slightly less than rewarding TPs:

\[
\min \frac{N-1}{N} \cdot \sum_{\forall n: y_n=0} y'_n - \sum_{\forall n: y_n=1} y'_n \tag{12}
\]

3. The third ILP formulation penalizes the number of used literals:

\[
\min \frac{N+1}{N} \cdot \sum_{\forall n: y_n=0} y'_n - \sum_{\forall n: y_n=1} y'_n + \sum_{m=1}^{M} l_m + l'_m \tag{13}
\]

4. The fourth ILP formulation combines the effects of the second and third ILP formulations:

\[
\min \frac{N-1}{N} \cdot \sum_{\forall n: y_n=0} y'_n - \sum_{\forall n: y_n=1} y'_n + \sum_{m=1}^{M} l_m + l'_m \tag{14}
\]

Objective functions (13) and (14) aim to counteract overfitting. Objective functions (12) and (14) include more TPs at the expense of more FPs, where additional FPs are ideally ignored or correctly classified afterwards by using the classification options described in the next section. The goal is to increase accuracy and recall while not decreasing precision in the best case.

For simplification purposes, we treat the effect of ILP formulation 1 as “not including more TPs at the expense of more FPs”, although a correct description would be “including less FPs at the expense of less TPs”. The effect of ILP formulation 3 is penalizing the number of used literals. Table 3 gives a summarizing overview of our ILP formulations for LOCATOR.

<table>
<thead>
<tr>
<th>Include more TPs at the expense of more FPs</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Penalize the number of used literals</td>
<td>No</td>
<td>ILP 1</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>ILP 3</td>
</tr>
</tbody>
</table>

Table 3: Overview of ILP formulations for LOCATOR and their effects. Note that ILP formulation is abbreviated to “ILP”.

### 3.2.2 Classification Options

After constructing a logic model, we can use it to classify datapoints with unknown class label. For classification, we use data on which dimensionality reduction was not applied. Let \( T \) be a set that contains each label of a class whose DNF formula is true for a datapoint \( X_n \). If \( |T| = 0 \), we do not label \( X_n \). If \( |T| = 1 \), we label \( X_n \) as \( c \in T \). However, if \( |T| > 1 \), we can select one among the five classification options and their combinations below. Let \( C \) be the number of classes in the data, and let \( \mathbf{x} \) be a \( C \times 1 \) vector where an entry \( x_c \) is the average number of present features in datapoints of class \( c \). Let \( x_n \) be the number of present features in \( X_n \). The five main classification options read as follows:
1. Do not label $X_n$.

2. Label $X_n$ as $c' = \min_{c \in T} \{|x_n - \bar{x}_c|\}$, i.e., label $X_n$ as the class with a true DNF formula for $X_n$ and with the closest average number of present features in datapoints. In case of a draw, do not label $X_n$.

3. Label $X_n$ as $c' = \min_{c \in T} \left\{ \frac{|x_n - \bar{x}_c|}{\sigma_c} \right\}$, where $\sigma_c$ is the estimated standard deviation of the number of present features in datapoints of class $c$, which can be calculated from the data. In other words, label $X_n$ as the class with a true DNF formula for $X_n$, while assuming that a normal distribution underlies the number of present features in datapoints of each class. In case of a draw, do not label $X_n$.

4. For each $c \in T$, save the first appearance of a true minterm in the corresponding DNF formula. Let $c'$ be the class with the earliest appearance. Label $X_n$ as $c'$ if there is exactly one such $c'$. In case of a draw, do not label $X_n$.

5. For each $c \in T$, count the number of true minterms in the corresponding DNF formula. Let $c'$ be the class with the most true minterms. Label $X_n$ as $c'$ if there is exactly one such $c'$. In case of a draw, do not label $X_n$.

Note that for simplicity, we assume that classification options 2 and 3 never result in a draw.

The idea behind classification option 2 is that we can correctly classify tumor DNA samples with ambiguous class based on the assumption that different cancer types have different average numbers of mutated genes.

Classification option 3 is very similar to classification option 2, the only difference being the division by $\sigma_c$. In statistics, the $Z$-score is the number of (estimated) standard deviations by which the datapoint value ($x_n$) is above or below the mean value ($\bar{x}_c$). We are not interested in whether $x_n$ is above or below $\bar{x}_c$, but in the distance between them, i.e., we take the absolute value of $x_n - \bar{x}_c$. Hence, the modified $Z$-score of class $c$, which we denote as $Z_c$, is calculated by

$$Z_c = \frac{|x_n - \bar{x}_c|}{\sigma_c}.$$ 

The smaller $Z_c$ is, the more likely $X_n$ belongs to class $c$.

As a reminder, a minterm of a LOCATOR model is supposed to represent a driver pathway.

Classification option 4 considers the order of minterms in a DNF formula. Roughly speaking, the more tumor DNA samples of the same cancer type share a similar driver pathway, the earlier the representing minterm appears in the corresponding DNF formula. The idea behind this classification option is that a tumor DNA sample is more likely to follow a driver pathway represented by an earlier appearing minterm.

Classification option 5 considers the number of true minterms, where we assume that the more likely a tumor DNA sample belongs to a cancer type, the more driver pathways of this cancer type it follows.
In case of a draw, classification options 4 and 5 do not classify datapoints such that we can use another classification option afterwards. Next, we present all possible combinations of the classification options mentioned above by arranging the digits of involved main classification options in a sequence. The order of the sequence determines the order of applying the classification options on a datapoint $X_n$, e.g., the sequence “4-2” means that we first apply classification option 4, and if $X_n$ is still not classified, we apply classification option 2. The ten combinations of the main classification options read as follows:

6. 4-2, 11. 5-2,
7. 4-3, 12. 5-3,
8. 4-5, 13. 5-4,
9. 4-5-2, 14. 5-4-2,
10. 4-5-3, 15. 5-4-3.
3.3 CATMEME

Figure 3 shows the CATMEME workflow, which can be briefly described as follows: CATMEME requires data where the last column stores class labels and the remaining columns are features that are either present (= 1) or absent (= 0). Next, the dimensionality of the data is reduced with SVM-RFE as described in Section 3.1. Afterwards, a set of potential class-specific, mutually exclusive features is established for each class. Together, these sets form a model that can eventually classify datapoints with unknown class label.

![CATMEME workflow diagram]

Figure 3: CATMEME workflow. Note that only the core components are displayed.

3.3.1 Establishing Sets of Class-Specific, Mutually Exclusive Features

In this section, we present different CATMEME variants to establish sets of class-specific, mutually exclusive features. Let $N$ be the number of datapoints and let $M$ be the number of features. For each class $c$, the set of potential class-specific, mutually exclusive features is constructed by an ILP that needs the following inputs:

1. $X$, an $N \times M$ binary matrix where an entry $X_{nm}$ is 1 if feature $X_m$ is present in datapoint $X_n$.
2. $y$, an $N \times 1$ binary vector where an entry $y_n$ is 1 if datapoint $X_n$ belongs to class $c$.

Note that $y$ can be easily created with the column storing the class labels.

The ILP formulations for CATMEME search for a set of potential class-specific, mutually exclusive features for the data such that the penalty for class-specificity and mutual exclusivity is minimal. This penalty and its condition (when to apply it) vary between the ILP formulations.

Given $X$ and $y$, the ILP for establishing a set $\phi$ of class-specific, mutually exclusive features for a class $c$ employs two sets of variables:

$$s_m = \begin{cases} 1 & \text{if feature } X_m \text{ is in } \phi, \\ 0 & \text{else.} \end{cases}$$

$$p_n = \begin{cases} 1 & \text{if class-specific mutual exclusivity for class } c \text{ is violated for datapoint } X_n, \\ 0 & \text{else.} \end{cases}$$

We describe how to formally and correctly formulate the ILP in Section 3.3.2. Here, we make use of the Iverson bracket for better understanding. The Iverson bracket for a condition $\Psi$ is defined as:

$$[\Psi] = \begin{cases} 1 & \text{if } \Psi \text{ is true}, \\ 0 & \text{else.} \end{cases}$$
1. The simplified formulation of the first ILP formulation reads as follows:

\[
\min \sum_{n=1}^{N} p_n \quad (15)
\]

s. t. \( p_n \geq \left[ \sum_{m=1}^{M} X_{nm} \cdot s_m \neq 1 \right] \quad \forall n : y_n = 1 \quad (16)\)

\[p_n \geq \left[ \sum_{m=1}^{M} X_{nm} \cdot s_m = 1 \right] \quad \forall n : y_n = 0 \quad (17)\)

\[p_n \in \{0, 1\} \quad n = 1, \ldots, N \quad (18)\)

\[s_m \in \{0, 1\} \quad m = 1, \ldots, M \quad (19)\)

The objective function (15) aims at minimizing the sum over all penalties \(p_n\). Constraints (16) ensure for each datapoint \(X_n\) of class \(c\) that \(p_n\) is 0 if exactly one feature is present and hence mutually exclusive for class \(c\). On the contrary, constraints (17) ensure for every other datapoint \(X_n\) of a class other than \(c\) that \(p_n\) is 1 if exactly one feature is present and hence mutually exclusive for a class other than \(c\). For this ILP formulation, all variables \(s_m\) that are set to 1 can be connected by logical XOR-gates such that a logical formula is constructed that can be used for classification.

CATMEME offers the following additional ILP formulations:

2. The second ILP formulation replaces constraints (16):

\[p_n \geq \left( \sum_{m=1}^{M} X_{nm} \cdot s_m \geq 2 \right) \quad \forall n : y_n = 1 \quad (20)\]

We mitigate the penalty condition of constraints (16) and do not penalize datapoints with zero present features in \(\phi\) because these datapoints do not add any information about mutual exclusivity, but they do not violate it either.

3. The third ILP formulation replaces constraints (16) and (18):

\[p_n \geq (\sum_{m=1}^{M} X_{nm} \cdot s_m) - 1 \quad \forall n : y_n = 1 \quad (21)\]

\[p_n \in \mathbb{N}_0 \quad n = 1, \ldots, N \quad (22)\]

As a consequence, if mutual exclusivity is violated, we add a penalty equal to the number of present features in a datapoint minus one.

4. The fourth ILP formulation also replaces constraints (16) and (18):

\[p_n \geq (\sum_{m=1}^{M} X_{nm} \cdot s_m) - 1 \quad \forall n : y_n = 1 \quad (23)\]

\[p_n \geq - (\sum_{m=1}^{M} X_{nm} \cdot s_m) + 1 \quad \forall n : y_n = 1 \quad (24)\]

\[p_n \in \mathbb{N}_0 \quad n = 1, \ldots, N \quad (25)\]
Therefore, the fourth ILP formulation combines the first and third ILP formulations, meaning that it works like the third, but also adds a penalty for zero present features like the first.

5. The fifth ILP formulation replaces constraints (17):

\[
p_n \geq \left[ \sum_{m=1}^{M} X_{nm} \cdot s_m \right] \geq 1 \quad \forall n : y_n = 0
\]

This means we penalize datapoints of a class other than \( c \) that have any present feature because we want \( \Phi \) to be specific for class \( c \).

6. The sixth ILP formulation combines the second and fifth ILP formulations and thus replaces constraints (16) and (17) accordingly.

7. The seventh ILP formulation combines the third and fifth ILP formulations and thus replaces constraints (16), (17), and (18) accordingly.

8. The eighth ILP formulation combines the fourth and fifth ILP formulations and thus replaces constraints (16), (17), and (18) accordingly.

Table 4 gives a summarizing overview of the penalties and their conditions of our ILP formulations for CATMEME.

<table>
<thead>
<tr>
<th>ILP formulation</th>
<th>penalty ( y_n = 0 )</th>
<th>( y_n = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( F = 0 )</td>
<td>( F = 1 )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\( F = \sum_{m=1}^{M} X_{nm} \cdot s_m \)

Table 4: Overview of ILP formulations for CATMEME and their penalties depending on which conditions hold true for \( y_n \) and \( F \), where \( F = \sum_{m=1}^{M} X_{nm} \cdot s_m \). Each entry is a penalty value. Note that ILP formulations 1 to 4 are identical to ILP formulations 5 to 8 if we disregard the fourth column (\( y_n = 0, F \geq 2 \)).
3.3.2 Detailed ILP Formulations

Constraints (16), (17), (20), and (26) from the previous section make use of the Iverson bracket, for which we now show the formally correct formulation.

Roughly speaking, the main idea behind all following reformulations is to use a sufficiently large number to force a binary variable to be either 1 or 0 for satisfying an inequality. As each Iverson bracket depends on \( \sum_{m=1}^{M} X_{nm} \cdot s_m \), which sums up to at most \( M \), we use \( M \) as our sufficiently large number.

- We reformulate constraints (16) \((\sum_{m=1}^{M} X_{nm} \cdot s_m \neq 1)\) as:
  \[
  \sum_{m=1}^{M} X_{nm} \cdot s_m - p_n \cdot M \leq 1 \quad \forall n : y_n = 1
  \]
  \( (27) \)
  \[
  \sum_{m=1}^{M} X_{nm} \cdot s_m + p_n \cdot M \geq 1 \quad \forall n : y_n = 1
  \]
  \( (28) \)

  If \( \sum_{m=1}^{M} X_{nm} \cdot s_m \geq 2 \), constraints (27) force \( p_n \) to be 1. If \( \sum_{m=1}^{M} X_{nm} \cdot s_m = 0 \), constraints (28) force \( p_n \) to be 1. If \( \sum_{m=1}^{M} X_{nm} \cdot s_m = 1 \), \( p_n \) can be 0 or 1, but it is set to 0 because we minimize \( \sum_{n=1}^{N} p_n \) in objective function (15).

- We reformulate constraints (17) \((\sum_{m=1}^{M} X_{nm} \cdot s_m = 1)\) as:
  \[
  \sum_{m=1}^{M} X_{nm} \cdot s_m - a_n \cdot M + \alpha'_n \leq 1 \quad \forall n : y_n = 0
  \]
  \( (29) \)
  \[
  \sum_{m=1}^{M} X_{nm} \cdot s_m - a_n \cdot M + \alpha'_n \cdot M \geq 1 \quad \forall n : y_n = 0
  \]
  \( (30) \)
  \[
  p_n + a_n + \alpha'_n = 1 \quad \forall n : y_n = 0
  \]
  \( (31) \)
  \[
  a_n \in \{0, 1\} \quad \forall n : y_n = 0
  \]
  \( (32) \)
  \[
  \alpha'_n \in \{0, 1\} \quad \forall n : y_n = 0
  \]
  \( (33) \)

  If \( \sum_{m=1}^{M} X_{nm} \cdot s_m \geq 2 \), constraints (29) force \( a_n \) to be 1 and thus \( p_n \) to be 0. If \( \sum_{m=1}^{M} X_{nm} \cdot s_m = 0 \), constraints (30) force \( \alpha'_n \) to be 1 and thus \( p_n \) to be 0. If \( \sum_{m=1}^{M} X_{nm} \cdot s_m = 1 \), constraints (29) and (30) force \( a_n \) and \( \alpha'_n \) to be 0, and thus constraints (31) force \( p_n \) to be 1.

- We reformulate constraints (20) \((\sum_{m=1}^{M} X_{nm} \cdot s_m \geq 2)\) as:
  \[
  \sum_{m=1}^{M} X_{nm} \cdot s_m - p_n \cdot M \leq 1 \quad \forall n : y_n = 1
  \]
  \( (34) \)

  If \( \sum_{m=1}^{M} X_{nm} \cdot s_m \geq 2 \), constraints (34) force \( p_n \) to be 1.

- We reformulate constraints (26) \((\sum_{m=1}^{M} X_{nm} \cdot s_m \geq 1)\) as:
  \[
  \sum_{m=1}^{M} X_{nm} \cdot s_m - p_n \cdot M \leq 0 \quad \forall n : y_n = 0
  \]
  \( (35) \)

  If \( \sum_{m=1}^{M} X_{nm} \cdot s_m \geq 1 \), constraints (35) force \( p_n \) to be 1.
4 Results

We implemented and ran LOCATOR and CATMEME using Python 3, Gurobi [11], and Snake-
make [12]. Gurobi is a math programming solver that includes a Python interface and is suitable
for solving our ILPs. Snakemake is a workflow management system that uses a Python-based
coding language to create scalable and reproducible data analyses. Workflows can be executed
on computer clusters with a job scheduling system, which allows for parallel execution of tasks.
This is especially useful for running a workflow with different settings simultaneously as well
as solving the ILPs for each class simultaneously.

The computational infrastructure on which we ran LOCATOR and CATMEME has the
following technical details: 2x Intel Xeon Gold 6136 (Skylake) with 3.00 GHz and 12 cores as
CPU, 192 GB RAM, and GPFS with approximately 3.5 PB, which is connected with 100 Gbit/s
InfiniBand.

For constructing a LOCATOR or CATMEME model for our dataset, we use up to 24 CPUs,
32 GB RAM, and a wall time of two days for each class on our used computational infrastruc-
ture. SVM-RFE needs 1 CPU, 8 GB RAM, and a wall time of one day. Almost all ILPs were
solved to optimality within the allotted wall time, and the few exceptions are mentioned in Sec-
tion 4.2. Furthermore, we use fixed random seeds in our implementations of LOCATOR and
CATMEME for reproducibility, because k-fold cross-validation and SVMs include randomness,
and ILPs can also have multiple optimal solutions, of which Gurobi chooses one at random.

4.1 LOCATOR

Soh et al.’s SVM-RFE [1] uses \( \frac{1}{4} \) of the datapoints as the test set while preserving the per-
centages of each class. Their evaluation metrics include accuracy, average precision and recall
(over all classes), precision as well as recall. Similarly, we evaluate LOCATOR with stratified
4-fold cross-validation, using the same five evaluation metrics, and compare it to SVM. We
used the StratifiedKFold tool from the Scikit-learn package [8] to split the data into four
folds. The dimensionality of each fold was reduced from 13150 to approximately 220 features.
This number of features may slightly vary for each fold, depending on the random splitting of
StratifiedKFold and the random initialization of LinearSVC for RFE.

In order to assess LOCATOR, we directly compare LOCATOR with SVM by evaluating
an SVM model on the same four folds with reduced dimensionality with the same five eval-
uation metrics. Therefore, we constructed an SVM model by running Soh et al.’s SVM-RFE
implementation without the code for RFE. Their implementation constructs multiple models
with different regularization parameters \( C \) and keeps the best. In our case, the model with
\( C = 0.02 \) yields the best results. Furthermore, we compare the results of LOCATOR with the
results of Soh et al.’s SVM-RFE as reported in [1]. Note that Soh et al.’s SVM-RFE is different
from the one described in Section 3.1 for two reasons. First, they used SVM-RFE not only for
dimensionality reduction, but also for constructing their SVM model. Second, they kept 900
features such that the highest possible accuracy for linear SVMs is achieved.
4.1.1 Accuracy, Precision, and Recall

In this section, we evaluate all 16 LOCATOR variants with unlimited $L$ and $P$, where each variant is a combination of an ILP formulation and up to one modification that excludes features, and compare LOCATOR to SVM and Soh et al.’s SVM-RFE.

Figure 4 shows the accuracy, average precision, and average recall of all LOCATOR variants with all classification options.

In the further course, we refer to the $i$th ILP formulation as “ILP $i$”.

In general, accuracy and recall seem to have an extremely strong correlation. Hence, we occasionally investigate only precision and recall while leaving out accuracy. Moreover, the results mostly exhibit the typical trade-off between precision and recall.

In terms of precision, ILP 3 has the best performance, although ILP 4 comes close. Among the usage of modifications, there is no clear winner as the results vary for each ILP formulation. In terms of recall, ILP 4 performs best among all alternatives. The same goes for using no modification. Besides, the performances of modifications B and C are almost identical to each other.

In terms of precision, all classification options perform similarly but classification option 1, which wins by a landslide. In terms of recall, however, classification option 1 is by far the worst, while the other options again perform similarly.

For the rest of Section 4.1.1, we mainly focus on evaluating the two overall best performing LOCATOR variants. We write a LOCATOR variant as LOCATOR $i$ for the combination of ILP $i$ and no modification. In terms of precision, the best variant is LOCATOR 3. In terms of recall, the best variant is LOCATOR 4. In addition, we focus on classification option 1, which achieves the highest precision with LOCATOR 3, and classification option 5-2, which achieves the highest recall with LOCATOR 4. In the further course, we denote LOCATOR $i$ with classification option $j$ as LOCATOR $ij$, e.g., we denote LOCATOR 3 or 4 with classification option 1 or 5-2 as LOCATOR 3$_1$ or LOCATOR 4$_{5-2}$, respectively.

Figure 5 shows precision and recall of SVM, Soh et al.’s SVM-RFE, and LOCATOR 3$_1$ and 4$_{5-2}$.

The most prominent difference is that LOCATOR 3$_1$ and LOCATOR 4$_{5-2}$ achieve low recall in comparison to SVM and Soh et al.'s SVM-RFE. However, the proportions between the classes are similar. For example, all approaches achieve low precision and/or recall for classes 11 (AdCC), 13 (CESC), 24 (SARC), and 26 (UCS) as compared to the other classes. Still, LOCATOR 3$_1$ and LOCATOR 4$_{5-2}$ have especially bad results for class 11 and/or class 26. However, the extremely low precision and/or recall might not be fully comparable to SVM and Soh et al.’s SVM-RFE. For LOCATOR, we set the precision or recall of a class to 0 if there is a division by zero, see Section 2.6, and this division by zero does not occur for SVM and Soh et al.’s SVM-RFE.

Table 5 lists accuracy, average precision, and average recall including their standard deviations of LOCATOR 3$_1$, LOCATOR 4$_{5-2}$, SVM, and Soh et al.'s SVM-RFE.
Figure 4: (a) Accuracy, (b) average precision, and (c) average recall of all 16 LOCATOR variants per classification option. The y-axis shows the evaluation metric. The x-axis shows the 15 classification options from Section 3.2.2. Note that modification is abbreviated to “mod”. Both $L$ and $P$ were not limited. The results were obtained via stratified 4-fold cross-validation.
Figure 5: Precision and recall of (a) SVM with $C = 0.02$, (b) Soh et al.’s SVM-RFE (adapted from [1]), (c) LOCATOR 3$^1$, and (d) LOCATOR 4$^{5,2}$. The y-axis shows classes, i.e., cancer types. The blue bars show precision, the orange bars show recall. Both $L$ and $P$ were not limited for (c) and (d). The results were obtained via stratified 4-fold cross-validation, except for (b) as they are adapted.

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Average precision</th>
<th>Average recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCATOR 3$^1$</td>
<td>62.8 % ± 1.0</td>
<td>84.6 % ± 2.8</td>
<td>56.1 % ± 1.5</td>
</tr>
<tr>
<td>LOCATOR 4$^{5,2}$</td>
<td>69.7 % ± 1.3</td>
<td>79.9 % ± 1.5</td>
<td>63.6 % ± 1.7</td>
</tr>
<tr>
<td>SVM</td>
<td>86.6 % ± 1.1</td>
<td>88.0 % ± 2.0</td>
<td>84.0 % ± 1.6</td>
</tr>
<tr>
<td>Soh et al.’s SVM-RFE</td>
<td>88.4 % ± 0.2</td>
<td>88.0 % ± 2</td>
<td>84.0 % ± 2</td>
</tr>
</tbody>
</table>

Table 5: Accuracy, average precision, and average recall of LOCATOR 3$^1$, LOCATOR 4$^{5,2}$, SVM with $C = 0.02$, and Soh et al.’s SVM-RFE. The values after ± denote the standard deviations. Note that for Soh et al.’s SVM-RFE, the standard deviations are not fully comparable due to a different experimental set-up as described in Section 4.1. Both $L$ and $P$ were not limited for LOCATOR 3$^1$ and LOCATOR 4$^{5,2}$. The results were obtained via stratified 4-fold cross-validation, except for Soh et al.’s SVM-RFE.
LOCATOR 3 has 23.8 % lower accuracy, 3.4 % lower average precision, and 27.9 % lower average recall than SVM, while LOCATOR 4-2 has 16.9 % lower accuracy, 8.1 % lower average precision, and 20.4 % lower average recall than SVM. However, if we disregard classes 11 and 26 for both SVM and LOCATOR because of the divisions by zero, the gaps for average precision and recall close a lot, whereas the gap for accuracy widens a little. Particularly, accuracy, average precision, and average recall without classes 11 and 26 are 91.7 %, 87.1 %, and 85.5 % for SVM, 66.7 %, 88.4 %, and 60.0 % for LOCATOR 3, and 73.8, 82.4 %, and 67.3 % for LOCATOR 4-2, respectively.

For evaluating the classification options in a more detailed manner, we computed the expected accuracy when classifying a datapoint with ambiguous class randomly as one of the classes that come into question, i.e., we computed one divided by the average number of true DNF formulas for those datapoints. For LOCATOR 3, we have 2.06 true DNF formulas on average, and thus an expected accuracy of $\frac{1}{2.06} = 48.5 \%$. For LOCATOR 4, we have 2.11 true DNF formulas on average, and thus an expected accuracy of $\frac{1}{2.11} = 47.4 \%$. Note that for simplicity, we assumed that the correct class is always among those classes that come into question. Table 6 lists the accuracy values including their standard deviations of all classification options but the first for LOCATOR 3 and 4.

<table>
<thead>
<tr>
<th>Classification option</th>
<th>LOCA TOR 3</th>
<th>LOCA TOR 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>51.6 % ± 3.8</td>
<td>52.5 % ± 2.0</td>
</tr>
<tr>
<td>3</td>
<td>54.4 % ± 2.0</td>
<td>54.0 % ± 1.8</td>
</tr>
<tr>
<td>4</td>
<td>49.7 % ± 1.4</td>
<td>51.2 % ± 5.3</td>
</tr>
<tr>
<td>5</td>
<td>52.2 % ± 3.7</td>
<td>53.8 % ± 5.9</td>
</tr>
<tr>
<td>4-2</td>
<td>55.5 % ± 3.5</td>
<td>57.6 % ± 3.2</td>
</tr>
<tr>
<td>4-3</td>
<td>49.7 % ± 1.4</td>
<td>51.2 % ± 5.3</td>
</tr>
<tr>
<td>4-5</td>
<td>56.4 % ± 1.5</td>
<td>57.6 % ± 3.0</td>
</tr>
<tr>
<td>4-5-2</td>
<td>59.3 % ± 3.6</td>
<td>61.4 % ± 1.7</td>
</tr>
<tr>
<td>4-5-3</td>
<td>56.4 % ± 1.5</td>
<td>57.6 % ± 3.0</td>
</tr>
<tr>
<td>5-2</td>
<td>62.3 % ± 4.2</td>
<td>62.3 % ± 1.7</td>
</tr>
<tr>
<td>5-3</td>
<td>52.2 % ± 3.7</td>
<td>53.8 % ± 5.9</td>
</tr>
<tr>
<td>5-4</td>
<td>56.4 % ± 1.5</td>
<td>57.6 % ± 3.0</td>
</tr>
<tr>
<td>5-4-2</td>
<td>59.3 % ± 3.6</td>
<td>61.4 % ± 1.7</td>
</tr>
<tr>
<td>5-4-3</td>
<td>56.4 % ± 1.5</td>
<td>57.6 % ± 3.0</td>
</tr>
</tbody>
</table>

Table 6: Accuracy for all datapoints with ambiguous class of each classification option with LOCATOR 3 and 4. The values after ± denote the standard deviations. The last row shows the expected accuracy when classifying a datapoint with ambiguous class randomly as one of the classes that come into question. Note that for simplicity, we assumed that the correct class is always among those classes that come into question.
Interestingly, although the expected accuracy for LOCATOR 4 is lower, almost all classification options achieve a slightly higher accuracy for LOCATOR 4 than for LOCATOR 3. However, note that some classification options achieve the same accuracy such that only eight out of the 14 accuracy values are distinct. All in all, our classification options yield acceptable, but not excellent results for classifying datapoints with ambiguous class since they do not perform much better than classifying randomly.

4.1.2 Limited Number of Literals and Minterms

Besides opposing overfitting, the options to limit $L$ and/or $P$ are supposed to aid in detecting only the most important features responsible for each cancer type, which could also enable better interpretability as the DNF formulas will probably consist of fewer minterms and literals. Furthermore, lower $P$ and especially lower $L$ decrease the wall time required for LOCATOR.

Figure 6 shows the accuracy, average precision, and average recall of all LOCATOR variants with all classification options, where we limited both $L$ and $P$ to 10.

Again, accuracy and recall seem to have an extremely strong correlation and the results mostly exhibit the typical trade-off between precision and recall. Compared to unlimited $L$ and $P$, the performances of the classification options are relatively similar. Furthermore, the results for the modifications are relatively similar as well. However, the results for ILP formulations 1 and 2 improve a lot, with LOCATOR 1$_{5.4}$ even outperforming LOCATOR 4$_{5.2}$ in terms of accuracy and recall. LOCATOR 1$_{5.4}$ achieves accuracy, precision, and recall values of 71.7% ± 0.8, 77.7% ± 0.9, and 65.6% ± 1.3, respectively. On the other hand, the results for ILP formulations 3 and 4 do not differ much, with the precision values staying about the same.

Hence, limiting both $L$ and $P$ can counteract overfitting for ILP formulations 1 and 2, but not for ILP formulations 3 and 4. However, if $L$ and/or $P$ are set too small, the results of ILP formulations 1 and 2 worsen again, see Figure 7, which shows the accuracy, average precision, and average recall of all LOCATOR variants with all classification options, where we limited $L$ to 10 and $P$ to 5.
Figure 6: (a) Accuracy, (b) average precision, and (c) average recall of all 16 LOCATOR variants per classification option. The y-axis shows the evaluation metric. The x-axis shows the 15 classification options from Section 3.2.2. Note that modification is abbreviated to “mod”. Both $L$ and $P$ were limited to 10. The results were obtained via stratified 4-fold cross-validation.
Figure 7: (a) Accuracy, (b) average precision, and (c) average recall of all 16 LOCATOR variants per classification option. The y-axis shows the evaluation metric. The x-axis shows the 15 classification options from Section 3.2.2. Note that modification is abbreviated to “mod”. \( L \) was limited to 10 and \( P \) was limited to 5. The results were obtained via stratified 4-fold cross-validation.
4.2 CATMEME

Because CATMEME focuses on establishing sets of potential class-specific, mutually exclusive features, we applied all eight ILP formulations of CATMEME on the whole dataset without cross-validation. In the further course, we denote a CATMEME variant using ILP \( i \) as CATMEME \( i \). The dimensionality of the dataset was reduced from 13150 to 250. Note that this number may vary slightly, depending on the random initialization of LinearSVC.

We mainly evaluate CATMEME based on the CGC mentioned in Section 3.1 instead of accuracy, precision, and recall. The CGC does not differentiate between SPMs and CNAs. Therefore, we disregard amplifications and deletions for CNAs and undo the subsumption of columns, which we described in Section 3.1. This resulted in 163 SPMs as well as 92 CNAs. There are 576 known driver mutations listed in the CGC, of which 62 appear in our dataset with reduced dimensionality. Furthermore, 53 of these 62 driver mutations appear in our 163 SPMs, while 12 of these 62 driver mutations appear in our 92 CNAs. Note that some SPMs and CNAs occur in the same genes, which is why 53 and 12 do not add up to 62.

We were not able to run CATMEME 1 on our used computational infrastructure within the allotted wall time for classes 1, 10, and 19, with the optimality gaps being 69.3 %, 16.2 %, and 41.2 %, respectively. Hence, we leave out the associated results. All other results of CATMEME are listed in Table 7, where for each class, we used the list of known driver mutations that we mention in Section 3.1. Note that we filtered out all mutations that do not appear in our dataset with reduced dimensionality.

The CATMEME variants 2, 3, 6, and 7 found no sets of potential class-specific, mutually exclusive features for any of the 28 classes, whereas CATMEME 1 found 11, CATMEME 4 found 10, CATMEME 5 found 8, and CATMEME 8 found 8 out of 28 of those sets.

In the following, we only consider CATMEME 1 because it finds the most known driver mutations with 12 out of 62. Speaking in terms of averages, CATMEME 1 finds 0.43 out of 4.82 known driver mutations for each class. The other mutations listed in the fifth column of Table 7 could either be yet unknown driver mutations or purely coincidental results. We further discuss this topic in Section 5.2, including the objective values in Table 7.

For classifying datapoints with these sets of class-specific, mutually exclusive features found by CATMEME 1, we construct a logical formula for each class by connecting all features in the corresponding set with logical XOR-gates. We can then check whether it is true or false for a datapoint, and use classification option 1, 2, or 3 from Section 3.2.2 if necessary. When using the whole dataset for constructing the CATMEME model as well as for evaluation, accuracy, average precision, and average recall are 25.0 %, 29.7 %, and 23.2 % for classification option 1, respectively. Figure 8 shows the corresponding precision and recall.

Because of these poor results for CATMEME 1, which is the best performing variant, we will not elaborate on cross-validation or classification for other variants. However, note that unlike CATMEME 1, the other variants need another method for classification, i.e., classification is not possible with logical formulas that use only logical XOR-gates.
<table>
<thead>
<tr>
<th>Class</th>
<th>ILP Objective value</th>
<th>Number of datapoints in class</th>
<th>Number of known driver mutations</th>
<th>Number of features in set</th>
<th>Overlap between set and known driver mutations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>862</td>
<td>973</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>5, 8</td>
<td>764</td>
<td></td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1, 4</td>
<td>193</td>
<td>280</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>1, 4, 5, 8</td>
<td>221</td>
<td>418</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1, 4, 5, 8</td>
<td>148</td>
<td>190</td>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>210</td>
<td>316</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>1, 4, 5, 8</td>
<td>142</td>
<td>279</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>2; 4</td>
<td>181; 199</td>
<td>231</td>
<td>3</td>
<td>106</td>
</tr>
<tr>
<td>16</td>
<td>1, 4, 5, 8</td>
<td>109</td>
<td>145</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>1, 4, 5, 8</td>
<td>230</td>
<td>332</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>216</td>
<td>399</td>
<td>5</td>
<td>95</td>
</tr>
<tr>
<td>21</td>
<td>1, 4, 5, 8</td>
<td>64</td>
<td>88</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>1, 4; 5, 8</td>
<td>62; 64</td>
<td>80</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 7: All results of CATMEME. The first column lists all classes for which the ILP formulations listed in the second column found a set of potential class-specific, mutually exclusive features. The third column lists the objective values of the corresponding ILP formulations per class. The fourth column lists the number of datapoints in the corresponding class. The fifth column lists the number of known driver mutations per class. For each class, the sixth column lists the number of potential class-specific, mutually exclusive features in a set. The last column lists the overlap between a set of class-specific, mutually exclusive features and the class-specific known driver mutations. The results were obtained by applying each CATMEME variant on the whole dataset.

Figure 8: Precision and recall of CATMEME 1 with classification option 1. The y-axis shows classes, i.e., cancer types. The blue bars show precision, the orange bars show recall. The results were obtained by applying CATMEME 1 on the whole dataset.
5 Discussion

The following section critically analyzes the methods and results presented in this thesis.

5.1 Data and Preprocessing

For LOCATOR and CATMEME, constructing the DNF formula or establishing a set of potential class-specific, mutually exclusive features requires a different wall time depending on the class. On our computational infrastructure and for the dataset after dimensionality reduction, most classes require less than two hours, while some classes require several hours (classes 1 and 24) or two days (class 19) when running LOCATOR 3 or 4, or even more than a week (classes 1, 10, and 19) when running CATMEME 1. We suspect that class 1 requires a long wall time because it has the most datapoints by far. Furthermore, class 24 is sarcoma, i.e., a cancer type with many subtypes. The many possible primary sites for sarcoma might therefore be the reason for the long wall time required for constructing its DNF formula. Unfortunately, we were unable to find explanations for the long wall times required for classes 10 and 19.

Our SVM-RFE reduces the dimensionality from 13150 to approximately 250, meaning that the dataset probably contains many passenger mutations. Besides, all choices for our SVM-RFE are arbitrary, meaning that it is not necessarily applicable to other datasets as the dimensionality might not be sufficiently reduced to run LOCATOR and CATMEME in reasonable time.

Moreover, there are two issues concerning the CGC, which might have distorted some of the results. First, manually compiling the lists of known driver mutations for each class is prone to errors. Second, some of the mutations listed in the CGC are not present in our dataset.

5.2 LOCATOR

In this subsection, we try to identify possible reasons for the results of LOCATOR regarding modifications that exclude features, ILP formulations, and classification options. Moreover, we discuss the potential and limitations of LOCATOR for achieving the main goal, which is the construction of explainable models with high credibility and interpretability.

5.2.1 Modifications

In Section 3.2.1, we introduced three modifications that exclude features, i.e., they ban reusing certain literals. Here, we give a short recapitulation and then hypothesize based on their results.

Modification A bans reusing any literal. In general, using this modification yields much worse results than using no modification. There are multiple conceivable hypotheses, which do not exclude one another. If the results for a cancer type worsened because we banned a literal of a minterm $\mu$, this could mean that

(a) it was a negative literal that should be reused as such in another minterm, because it is a mutation that rarely occurs in this cancer type.
(b) it was a positive literal that should be reused as such in another minterm, because it is a mutation that is included in multiple driver pathways for this cancer type.

(c) it was (i) a positive literal \( l \) that should be reused as \( \neg l \) or (ii) a negative literal \( \neg l \) that should be reused as \( l \) in another minterm, because it is a driver mutation that is mutually exclusive with (i) negative or (ii) positive literals in \( \mu \).

Modification B bans reusing any positive literals. In general, using this modification yields slightly worse results than using no modification, but better results than modification A. Hence, hypotheses (a) and/or (c)(ii) are further supported because reusing negative literals seemed to improve our results.

Modification C enforces mutual exclusivity between two literals \( l_i \) and \( l_j \) if both were used together as \( l_i \) and \( \neg l_j \) in a minterm. The performances of modifications B and C are almost identical. Hence, hypothesis (c) is further supported, e.g., if a DNF formula \((l_i \land \neg l_j) \lor l_j\) of modification B yields the same results as a DNF formula \((l_i \land \neg l_j) \lor (l_j \land \neg l_i)\) of modification C for all datapoints, then \( l_i \) and \( l_j \) cannot be simultaneously present in any datapoint.

Besides, all hypotheses are supported by the fact that using no modification, i.e., not banning reusing any literal, yields the best results overall.

5.2.2 ILP Formulations

In Section 3.2.1, we introduced two effects and their four corresponding ILP formulations, see Table 3.

The idea behind the effect of including more TPs at the expense of more FPs is to ideally ignore or correctly classify additional FPs afterwards by using the classification options. In other words, the goal is to increase accuracy and recall while not decreasing precision in the best case. This is not achieved for any LOCATOR variant with ILP formulation 2 or 4 as compared to ILP formulation 1 or 3, because the increase of accuracy and recall is not as high as the decrease in precision, see Figure 4 from Section 4.1.1. All in all, this idea leads to more datapoints with ambiguous class, while the accuracy of each classification option stays nearly the same. We believe that our idea would yield good results if the classification options were more reliable.

The idea behind the effect of penalizing the number of used literals is to counteract overfitting. ILP formulations 3 and 4, which apply this idea, perform much better than the other two for unlimited \( L \) and \( P \) despite using far fewer literals, see Table 8.

As shown in Section 4.1.2, limiting \( L \) to 10 and \( P \) to 5 yields better results for ILP formulations 1 and 2 as compared to unlimited \( L \) and \( P \), which strongly suggests that overfitting is counteracted by fewer minterms and literals per minterm. However, the precision values for ILP formulations 1 and 2 with limited \( L \) and \( P \) are lower than those of ILP formulations 3 and 4 with limited or unlimited \( L \) and \( P \). This indicates that limiting \( L \) and \( P \) does not seem to counteract overfitting as much as penalizing the number of used literals. An additional advantage of the latter is that we do not need to find optimal settings for \( L \) and \( P \).
<table>
<thead>
<tr>
<th>ILP</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mod</td>
<td>- A B C</td>
<td>- A B C</td>
<td>- A B C</td>
<td>- A B C</td>
</tr>
<tr>
<td>literals</td>
<td>64.4 32.2 62.7 64.9</td>
<td>58.1 33.3 57.9 58.9</td>
<td>8.4 8.6 8.5 8.5</td>
<td>6.8 6.9 7.0 7.0</td>
</tr>
<tr>
<td>minterms</td>
<td>12.8 7.2 13.1 15.1</td>
<td>12.7 6.9 12.8 15.0</td>
<td>3.7 3.3 3.5 3.5</td>
<td>4.0 3.6 3.7 3.7</td>
</tr>
</tbody>
</table>

Table 8: Average number of minterms and literals per minterm for each LOCATOR variant with unlimited $L$ and $P$. Using no modification is denoted as “-”. Note that modification is abbreviated to “mod”. The results were obtained via stratified 4-fold cross-validation.

Still, we think that our penalty for the number of used literals can be further improved. Currently, we penalize a literal as much as rewarding a correctly classified datapoint, which means that each literal must be descriptive for at least one datapoint. However, classes with fewer datapoints might not be able to use enough literals for a good minterm. For example, a minterm that uses merely 21 out of 250 literals might perfectly describe a class with 20 datapoints, but ILP 3 would not consider this minterm as the best solution. Therefore, a parameter that adjusts the penalty depending on the numbers of both features and datapoints could further improve the results.

### 5.2.3 Classification Options

Our classification options from Section 3.2.2 consist of five main options and ten combinations of those. We do not discuss classification option 1 because there is no cancer-related idea behind it.

Broadly speaking, classification options 2 and 3 consider the average number of mutations of cancer types, where classification option 3 additionally assumes that a normal distribution underlies the number of mutations. The latter option often performs slightly better than the former, but overall they perform very similarly. Furthermore, all combined classification options that include classification option 2 perform better than the corresponding combined classification options that include classification option 3. Hence, we cannot safely assume that a normal distribution underlies the numbers of mutations for a cancer type, and therefore disregard classification option 3 and its combinations in the further course. Furthermore, as our results for the classification options from Section 4.1.1 are not significantly better than classifying datapoints with ambiguous class randomly, we believe that the average number of mutations of cancer types alone is not a decisive factor for classification. Perhaps these classification options would perform better for data that includes information about the cancer stage, which is an indicator of how far a cancer spread in a patient’s body.

Classification option 4 considers the order of minterms, which is determined by the number of tumor DNA samples with similar driver pathways. The earliest appearance of a true minterm in a DNF formula of a class is the decisive factor when choosing between two classes. In case of a draw, no class is chosen. The performances of options that include classification option 4 are better for ILP alternatives 1 and 2 than for ILP alternatives 3 and 4. This is because
the latter two alternatives lead to far fewer minterms on average such that draws are much more probable. Since we prefer DNF formulas with fewer literals and minterms because they are easier to interpret, we disregard classification option 4 and its combinations in the further course.

The last main classification option 5 considers how many minterms are true. The highest number of true minterms in a DNF formula of a class is the decisive factor when choosing between two classes. In case of a draw, no class is chosen. The performance of classification option 5 is not good overall, though classification option 5-2 performs much better than classifying datapoints with ambiguous class randomly for ILP alternatives 3 and 4. This implies that hypothesis (b) from Section 5.2.1 may be true for some, but not all mutations since the number of true minterms with different positive literals mean that multiple pathways exist for a cancer type.

In summary, the performances of our classification options for datapoints with ambiguous class vary a lot for all LOCATOR variants and do not seem to exhibit a certain structure. Furthermore, there is a lot of room for improvement regarding classification of datapoints with ambiguous class.

5.2.4 Explainability

Many machine learning approaches, like SVMs for example, lack explainability because they start with a random initialization of a model and from there, they find a local optimum for the input data, where this found local optimum also varies depending on the optimization method. In contrast, we can easily explain all minterms found by LOCATOR because each found minterm is globally optimal for its input data. However, this does not mean that LOCATOR necessarily finds the globally optimal DNF formula for a class, e.g., a globally optimal minterm could include many additional TPs at the cost of some additional FPs, but these additional TPs might have been found by the next minterm without including these additional FPs.

5.2.5 Credibility

As a reminder, one of the main goals of LOCATOR is to construct logic models for cancer classification with high credibility. Therefore, we think that precision, which shows how credible predictions are, plays a more important role for LOCATOR than recall and accuracy, because recall is a measure of quantity rather than quality and a high accuracy does not necessarily imply a good model. For example, consider a dataset with 90 datapoints of class 0, and 10 datapoints of class 1. A model that labels all datapoints as class 0 achieves an accuracy of 90 %, but fails to distinguish both classes.

As a first sanity check of whether the results of LOCATOR are purely coincidental, we ran LOCATOR 3\_1 on a random binary matrix with the same row and column totals as our data matrix, while leaving the last column with class labels untouched [13]. Stratified 4-fold cross-validation yields an accuracy of 0.01 %, an average precision of 0.2 %, and an average
recall of 0.004%. Hence, mutations in tumor DNA probably occur after some pattern, which LOCATOR seems to detect up to a certain extent.

Our LOCATOR variants achieve a high precision and low standard deviation values for all evaluation metrics, which is a strong indication for credibility. Particularly, LOCATOR 31 achieves a precision almost as high as that of linear SVMs, or even higher if we do not consider the two classes for which both approaches perform the worst.

Besides, we can get a first idea of how credible a LOCATOR model is by investigating its DNF formulas. For each class, Table 9 lists overlaps between positive literals found by LOCATOR 4 with unlimited L and P, where these positive literals correspond to mutations, and the class-specific known driver mutations from the CGC. Note that we ignore negative literals as they stand for the absence of mutations and thus cannot drive cancer. We chose LOCATOR 4 over all other variants because it yields the highest precision values when applying and evaluating on the whole dataset, even with different settings for L and P.

As a reminder, the number of known driver mutations that appear in our dataset with reduced dimensionality is 62, of which LOCATOR 4 outputs 24 as positive literals. Speaking in terms of averages, LOCATOR 4 outputs 1.14 out of 4.82 known driver mutations for each class. Besides the possibility that our manually compiled lists of known driver mutations contain errors, it seems that LOCATOR 4 misses many cancer-type specific driver mutations, which could be connected with the rather low recall values. However, this does not affect the credibility of LOCATOR 4 because its precision is rather high. Therefore, all found positive literals are likely cancer-related, which we discuss further in the next section.

5.2.6 Interpretability

Another main goal of LOCATOR is to construct logic models for cancer classification with high interpretability. Since each literal stands for the presence or absence of a certain mutation, the DNF formulas can be easily understood, and understanding results is indispensable for interpreting them. Besides, LOCATOR allows for limiting L and P, which can further facilitate interpretation for some LOCATOR variants.

In the following, we make first conceivable interpretations of the results listed in Table 9. One possible interpretation is to assume that the positive literals not listed in the CGC are yet unknown driver mutations. If a positive literal appears in a minterm where no other positive literals are known driver mutations, chances are that this positive literal is indeed a driver mutation. This hypothesis is further supported by the widely accepted hypothesis that driver mutations with similar cancer-driving effects tend to be mutually exclusive [2, 3]. If a positive literal appears in a minterm where one or more other positive literals are known driver mutations, it could be a yet unknown driver mutation that needs to interact with other driver mutations. This hypothesis would be in accordance with Martincorena et al.’s findings that one to ten mutations are needed to drive cancer [14]. Moreover, these interactions might be cancer-type specific and thus provide further insight into cancer development as well.
<table>
<thead>
<tr>
<th>Class</th>
<th>Number of known driver mutations</th>
<th>Number of positive literals</th>
<th>Overlap between positive literals and known driver mutations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>8</td>
<td>KDM6A</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>17</td>
<td>CDH1, PIK3CA, TP53</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>10</td>
<td>APC, BRAF, FBXW7, KRAS</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>8</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>VHL</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>8</td>
<td>CBFB, CEBPA, FLT3, NPM1, RUNX1</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>7</td>
<td>KRAS</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>7</td>
<td>TP53</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>6</td>
<td>PTEN</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>8</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>4</td>
<td>AXIN1</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
<td>6</td>
<td>KRAS</td>
</tr>
<tr>
<td>17</td>
<td>9</td>
<td>7</td>
<td>ERG, ETV1, ETV4, SPOP</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>15</td>
<td>BRAF, NRAS</td>
</tr>
<tr>
<td>19</td>
<td>4</td>
<td>22</td>
<td>CDH1</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>8</td>
<td>BRAF, NRAS, RET</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>7</td>
<td>-</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>8</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>24</td>
<td>10</td>
<td>18</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>7</td>
<td>-</td>
</tr>
<tr>
<td>26</td>
<td>2</td>
<td>14</td>
<td>TP53</td>
</tr>
<tr>
<td>27</td>
<td>6</td>
<td>2</td>
<td>GNA11, GNAQ</td>
</tr>
</tbody>
</table>

Table 9: Overlapping mutations between LOCATOR 4 and the CGC. The second column lists the number of class-specific known driver mutations in the CGC per class after filtering out those mutations that do not appear in the dataset. The third column lists the number of positive literals per class. The last column lists overlaps between positive literals, which correspond to mutations, and the class-specific known driver mutations per class. The results were obtained by applying LOCATOR 4 on the whole dataset with unlimited $L$ and $P$. 

33
Therefore, the hypothesis of sets of cancer-type specific, strictly mutually exclusive feature sets from Section 3.2.1 might be worth looking into. Note that these interpretations are especially likely if these positive literals are used in minterms whose corresponding ILPs have good objective values.

5.3 CATMEME

A central issue with every CATMEME variant is the fact that the sets found do not necessarily consist of only driver mutations, but passenger mutations as well. Even if the hypothesis that driver mutations with similar cancer-driving effects tend to be mutually exclusive was true, it would not exclude the possibility that driver and passenger mutations are mutually exclusive as well. Furthermore, most of the mutations found by CATMEME are not listed as known driver mutations in the CGC. However, these mutations could possibly be yet unknown driver mutations. To investigate this, we considered the objective values, which are calculated according to objective function (15), and the corresponding number of datapoints that are both listed in Table 7 as well. The latter equals the worst case that can be achieved with an empty set of features. Therefore, the closer the objective value is to the number of datapoints, the higher is the probability that the corresponding set of features was found coincidentally. Hence, the possibility that CATMEME finds yet unknown driver mutations seems unlikely.

Kim et al. [4] state that mutually exclusive pairs of mutations are observed within many cancer types, while some are common across multiple different cancer types, which would be consistent with our results. Moreover, this hypothesis is further supported by the fact that the CGC often lists multiple cancer types for one driver mutation.

Nevertheless, there is still the possibility that SVM-RFE removes class-specific driver mutations. In fact, 555 out of the 576 known driver mutations listed in the CGC appear in our dataset, while only 62 of these 555 remain after applying SVM-RFE. However, it is unlikely that the removed known driver mutations are class-specific because the accuracy of LinearSVC, which is calculated in every SVM-RFE step, does not seem to drop significantly without them.

In summary, the hypothesis that there exists a set of mutually exclusive driver mutations for each cancer type is not supported by our results.
6 Conclusions

In this thesis, we introduced two approaches, LOCATOR and CATMIME, with several variants to construct models for aiding in cancer research, diagnostics, and therapy. Unlike other machine learning approaches, both aim to construct explainable models with high credibility and interpretability. The basic idea of LOCATOR is to construct models consisting of logical formulas in disjunctive normal form, while the basic idea of CATMIME is to construct models consisting of logical formulas where all literals are connected with logical XOR-gates.

For the dataset we used, which consists of tumor DNA samples with somatic mutations, most LOCATOR variants, but none of the CATMIME variants achieve this aim to a certain extent. Therefore, we conclude for LOCATOR that logical formulas in disjunctive normal form are suited for cancer classification, whereas the results of CATMIME do not support the hypothesis of an existing set of mutually exclusive driver mutations that are responsible for developing a specific cancer-type.

For future work on this topic, both LOCATOR and CATMIME can be improved such that they can be applied on other datasets and/or problem formulations. For example, previous studies have classified cancer types with machine learning approaches based on gene expression signatures [15] or DNA methylation profiles [16], which we can imagine for LOCATOR as well. In practical terms, we could improve the runtime by formulating the problem not as an ILP, but as a boolean satisfiability problem (SAT). SAT solvers have become faster and more popular over the last years, and they might be suitable to be integrated into LOCATOR, which also concerns boolean logic. In theoretical terms, we envision new variants, classification options, or a new dimensionality reduction to further improve precision and especially recall.

As a conclusion, logic models constructed by LOCATOR have potential for aiding in cancer research, diagnostics, and therapy. On top of that, LOCATOR provides a solid basis for future extensions in order to exploit its full potential.

Data and code availability.

Data and code for LOCATOR is available at https://gitlab.cs.uni-duesseldorf.de/tran/locator.
Data and code for CATMIME is available at https://gitlab.cs.uni-duesseldorf.de/tran/catmeme.
7 References


