# Machine learning for analysing metabolic networks

*Lilli Johanna Freischem*

**MInf Project (Part 1) Report**
Master of Informatics
School of Informatics
University of Edinburgh

2021

# Abstract

Drugs can influence the metabolic system of biological cells by targeting enzymes which catalyse metabolic reactions. Since cancer cells are characterised by having an altered metabolism, it may be possible to reduce the spread of cancer without affecting healthy tissue by stopping reactions that are essential for the growth of cancer cells only.

The metabolic activity of a cell is represented by a metabolic network, describing the relationship between biochemical reactions and their metabolites. Metabolic networks are characterised well enough to construct and analyse mathematical models of their behaviour. This project explores how machine learning can help finding candidate reactions for drug targets when applied to reaction-based metabolic networks.

## Acknowledgements

I would like to thank my supervisor Diego Oyarzún for organising this project, providing invaluable feedback and for creating an incredibly welcoming atmosphere in his research group.

Next, I would like to thank Denise Thiel for guiding me through this project and for her constant support throughout the year. I really enjoyed our weekly meetings which enabled me to stay on track with my project.

Außerdem möchte ich mich bei meiner Familie für ihre ununterbrochene Unterstützung während meines Studiums bedanken. Und besonders bei Mia, die jede Seite dieser Arbeit mehrfach korrekturgelesen hat.

Lastly, I would like to thank my friends who make Edinburgh my home away from home and the EUBC for keeping me active and sane during lockdown.

# Table of Contents

# Acronyms

| | |
|---|---|
| **COBRA** | Constraint-Based Reconstruction and Analysis. 2, 3, 7, 14–16, 18, 20, 21, 37, 38 |
| **CV** | Cross-Validation. 21, 26–30, 38 |
| **FBA** | Flux-Balance Analysis. 1–3, 5–8, 14, 17, 18, 20, 21, 32, 37–40 |
| **GEM** | Genome-Scale Metabolic Model. 4, 5, 7, 14, 17, 18, 20, 32, 38, 39 |
| **LP** | Linear Programming. 5–7 |
| **MCL** | Markov Clustering. 9, 16 |
| **MFG** | Mass Flow Graph. 1–3, 7, 8, 14–25, 28, 32, 34, 37–39 |
| **MLP** | Multilayer Perceptron. 13, 25, 26, 28–31, 35, 36 |
| **pFBA** | parsimonious Flux-Balance Analysis. 6, 15, 17, 18 |
| **ReFeX** | Recursive Feature eXtraction. 10, 21, 23, 24, 26–30, 34, 38 |
| **RF** | Random Forest. 12, 25, 29, 30, 34, 35 |
| **RolX** | Role eXtraction. 9–11, 21, 23, 24 |
| **SVM** | Support Vector Machine. 13, 25–29 |

# Chapter 1

# Introduction

## 1.1 Motivation

Cancer is not just one disease, but a collection of disorders caused by malfunctions of the human metabolism and alterations in the cellular signaling pathways [1][2]. The emergence of biochemical and molecular biological tools thus led to a rise in studies of the metabolism of cancer cells in the last decades [3].

Drugs can influence a cell's metabolic system by targeting enzymes which catalyse metabolic reactions [1]. As the metabolism of cancer cells is altered, it may be possible to reduce the spread of cancer without affecting healthy tissue [4]. Hence, one aim of cancer research is to find reactions which are essential for the growth of cancer cells only, known as therapeutic targets [4][5][6].

Cell metabolism can be computationally represented by metabolic networks [7]. A metabolic network comprises of a set of metabolites intertwined by biochemical reactions. Genome-scale metabolic networks are well enough characterised to construct and analyse mathematical models of their behaviour [8]. One of such models are Mass Flow Graphs (MFGs), which are reaction-based metabolic graphs that take the biological and environmental context of the cell into account [9].

MFGs have been used to study how removing reactions from the metabolic network affects node connectivities with the aim to find therapeutic targets [9]. For this purpose, reaction nodes currently have to be deleted individually to analyse the resulting metabolic network. However, this approach considers only one experimental condition at a time and is computationally expensive; flux-balance analysis (FBA) must be performed individually for each candidate reaction [9].

We attempt to overcome this performance bottleneck by using machine learning to predict the essentiality of reactions using the original graph only. This requires extracting features that enable predicting the far-reaching effects of removing a reaction from the metabolic network. A first attempt was to use node connectivity measures such as PageRank. However, it was found that node connectivity and reaction essentiality are not correlated [10].

A promising approach for predicting reaction essentiality is to use node role analysis. Two nodes belong to the same role if they have similar structural behaviour [11]. Using node role analysis on MFGs could facilitate extracting information on the importance of reactions for the metabolic network [12].

This project will explore whether machine learning algorithms can use metabolic networks and node role analysis to predict reaction essentiality in order to identify drug targets for cancer treatment.

## 1.2 Research Question

This project investigates the following research question:

> Can machine learning algorithms predict the biological essentiality of metabolic reactions based on the properties of Mass Flow Graphs?

For this purpose, we developed MFGpy, a python package for the automatic generation of MFGs. MFGs are reaction-based, weighted metabolic graphs (see Chapter 2). They are well-suited for this project as we analyse reaction essentiality of cells under specific genetic conditions. MFGpy was used to compute and analyse the MFGs of five cancer cell lines. This included computing the essentiality of every reaction in the MFGs using FBA. We approached essentiality prediction as classification problem (see Chapter 4). Therefore, the FBA reaction essentiality values were grouped in either two or four classes before training the classification algorithm.

## 1.3 Achievements

In this section, I provide a summary of the main achievements of this project.

1. I developed MFGpy, a python package for the automatic generation, analysis and visualisation of MFGs from a COBRA model. This included implementing the theoretical algorithm proposed by Beguerisse et al. [9].
2. Using MFGpy, I computed the MFGs of five cancer cell lines. Then, I conducted a series of experiments on their MFGs to analyse the metabolic networks. I provide methods for this analysis in MFGpy.
    (a) I computed the FBA reaction essentiality of every reaction in the graphs which is used as ground-truth for later analyses.
    (b) I identified and analysed outliers emerging from the removal of single reactions from MFGs. Outliers are nodes with unusual changes in centrality that cannot be captured by flux analysis.
3. I implemented the theoretical algorithm proposed by Cooper et al. [12] for computing flow profiles of nodes in directed networks.
4. I used machine learning to predict reaction essentiality. I compared a number of machine learning algorithms across an array of input feature sets [13].

## 1.4 Structure of the Report

In **Chapter 1**, I introduce this project by presenting the motivation behind it, explaining the research question and summarising my achievements.

In **Chapter 2**, I summarise previous research that forms the basis of this project. I explain how machine learning facilitates cell metabolism analysis by introducing Constraint-Based Reconstruction and Analysis (COBRA) of metabolic networks. In addition, I provide the theoretical background for MFGs and node role analysis and explain their potential for this project.

In **Chapter 3**, I present MFGpy, the python package for the automatic generation and analysis of MFGs. I describe how COBRA models are applied to extract essential information for building MFGs and explain implementation decisions. Furthermore, I present MFGpy's functions for analysing MFGs.

In **Chapter 4**, I describe essentiality prediction as machine learning problem by providing an overview of the machine learning pipeline. I present the feature sets that were used as input to the machine learning algorithms and explain the training and testing procedure.

In **Chapter 5**, I present and evaluate the results of our experiments: FBA essentiality analysis, outlier analysis, node role analysis and essentiality prediction.

In **Chapter 6**, I critically assess the methods used in this project and discuss potential applications for MFGpy and essentiality prediction in cancer research. Finally, I outline directions for future work based on this project.

# Chapter 2

# Background

In this chapter, the background information that forms the basis of this project is presented. First, cell lines and genome-scale metabolic models are briefly introduced. Then, constraint-based modelling is explained. Finally, recent research in predicting node roles in large-scale networks is presented and connected to our work.

## 2.1 Cell Lines

Cell lines originate from primary cell cultures that were extracted directly from cells, tissue, or organs and can reproduce indefinitely [14]. Cell lines from various human cancer types have been established and are widely used in medical research, especially basic cancer research and drug discovery [15][16].

In this project, we work with five cancer cell lines from the NCI-60 panel, BT-549, HCT 116, K-562, MCF7 and OVCAR-5. BT-549 is a cell line of ductal carcinoma, a common type of breast cancer that starts in the milk ducts [17]. HCT 116 is a cell line of colorectal carcinoma, a cancer of the large intestine [18]. K-562 is an erythroleukemia cell line, a cancer of the body's blood-forming tissues [19]. MCF7 is a cell line of breast adenocarcinoma, a type of breast cancer that starts in mucus-producing glandular cells of the body [20]. OVCAR-5 is a cell line of epithelial ovarian carcinoma, the most common type of ovarian cancer [21].

## 2.2 Genome-Scale Metabolic Models

Genome-scale metabolic models (GEMs) are mathematical models of the metabolic network of a cell [7][8]. To reconstruct GEMs, either manually or with automatic reconstruction tools, a list of all biochemical reactions in the cell is assembled together with data on cellular boundaries, a biomass assembly reaction, and exchange reactions with the environment [22].

We define a metabolic network containing $n$ metabolites $X_i$ (for $i = 1,...,n$) and $m$ reactions as:

$$R_j : \sum_{i=1}^{n} \alpha_{ij} X_i \leftrightharpoons \sum_{i=1}^{n} \beta_{ij} X_i, \quad j = 1,...,m \tag{2.1}$$

where $\alpha_{ij}$ and $\beta_{ij}$ are the stoichiometric coefficients of metabolite $X_i$ in reaction $R_j$. This means, when reaction $R_j$ takes place it consumes $\alpha_{ij}$ and produces $\beta_{ij}$ molecules of metabolite $X_i$. The stoichiometric coefficients can be compiled into the $n \times m$ stoichiometric matrix $\mathbf{S}$; each entry $S_{ij} = \beta_{ij} - \alpha_{ij}$ denotes the total number of $X_i$ molecules produced ($S_{ij} > 0$) or consumed ($S_{ij} < 0$) by reaction $R_j$ (see Figure 2.1a, b).

## 2.3 Constraint-Based Reconstruction and Analysis

The core concept of modelling GEMs is the identification and mathematical definition of constraints [8]. They define realistic metabolic behaviour by constraining the availability of nutrients and other metabolites and are easier to identify than kinetic parameters such as metabolic fluxes and metabolite concentrations [23]. Hence, in addition to the stoichiometric matrix, upper and lower flux bounds of reactions have to be defined [8].

Analysing human GEMs has contributed to a better understanding of the biological mechanisms behind various diseases and plays a key role in drug discovery [24]. Reconstructed metabolic networks are stored in a variety of databases such as BiGG Models [25] and BioModels [26], where they can be downloaded for further analysis and research.

Yizhak et al. developed an algorithm (PRIME) for reconstructing cell-specific GEMs based on molecular and gene-expression data [4]. They used it to generate GEMs of the NCI-60 cancer cell lines, including GEMs of the five cell lines presented above which we analyse in this project.

## 2.4 Flux-Balance Analysis

Flux-Balance Analysis (FBA) is the main method for studying constraint-based metabolic networks [9]. FBA is a Linear Programming (LP) algorithm that takes the reconstruction of a GEM as input and computes a distribution of cellular fluxes based on two assumptions, optimality and steady state. Figure 2.1 shows an overview of the formulation of an FBA problem.

A system of linear equations is derived from the stoichiometric matrix $S$ and upper and lower reaction bounds (see Figure 2.1c). It is assumed that the cell has entered a steady state where metabolite concentrations are constant, so consumption and production fluxes of every metabolite cancel each other out [27].

Additionally, the cell's objective function $Z$ is defined (see Figure 2.1d). It encodes the cell's biological goal based on the assumption that evolution has optimised cells for a certain metabolic task. The cellular objective of cancer cells as well as bacteria is the optimisation of cell growth.

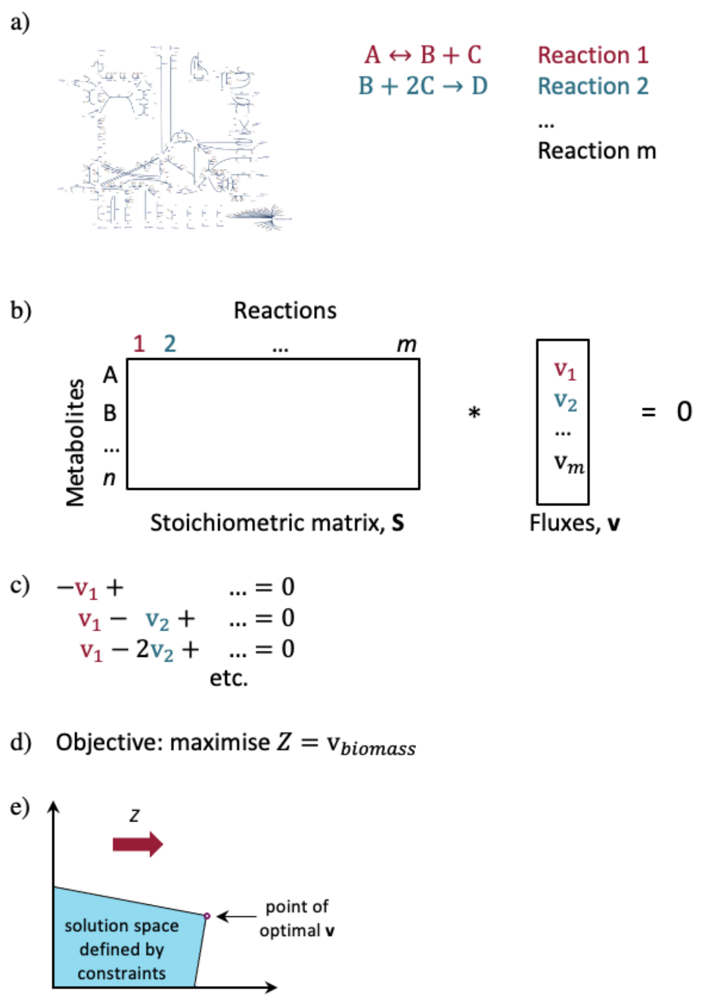

Figure 2.1: Formulation of an FBA problem. (a) A genome-scale metabolic network is reconstructed. (b) Metabolic reactions and constraints are mathematically represented. (c) A set of linear equations is defined by the mass balance. (d) An objective function $Z$ is defined. (e) Fluxes that maximise $Z$ are calculated. Figure adapted from [27].

Finally, LP is used to calculate a flux vector v which is a solution to the model's system of linear equations that maximises the objective function $Z$ (see Figure 2.1e).

By altering the constraints on a model, FBA can be used to simulate a cell's metabolic network under different environmental and genetic conditions [27].

A number of variants of the FBA algorithm are available, such as parsimonious Flux-Balance Analysis (pFBA). pFBA is a variant of FBA that minimises the energy required to optimise the objective function assuming that growing organisms favour cells which grow the fastest whilst requiring minimum energy [28].

## 2.5 COBRApy

With the rise of genome-scale metabolic network modelling, Becker et al. developed the COBRA toolbox, a leading software package providing methods for constraint-based generation and analysis of GEMs in MATLAB [29].

Based on the COBRA toolbox, Ebrahim et al. presented COBRA for Python (COBRApy) as part of the openCOBRA project, a community project aiming to improve the accessibility of and promote constraint-based research by making software freely available [30].

COBRApy is an object-oriented framework. Amongst others, it contains functionality for reading and writing COBRA models as SBML files and for performing FBA [30]. Additionally, COBRApy provides an interface to linear optimisation packages. This enables users to employ existing LP solvers such as GLPK [31] or Gurobi [32] for solving FBA problems. The object-oriented design of COBRApy makes information on reactions and metabolites of the GEM easily accessible [30].

## 2.6 Mass Flow Graphs

In addition to performing constraint-based analysis on GEMs, researchers analyse the structural properties of metabolic networks by applying tools from graph theory to graphs derived from GEMs [9]. However, the methods for turning metabolic models into networks pose challenges as the chosen graph construction strongly influences the information gained from the network [9]. Often, GEMs are analysed as undirected graphs. This ignores the directionality of flows although it is a key feature of metabolic reactions [12]. Further, the derived graphs often neglect environmental conditions and their influence on a cell's metabolic network [9].

Aiming to overcome these issues, Beguerisse-Diaz et al. developed a novel algorithm for deriving flux-based weighted graphs from organism-wide metabolic networks called Mass Flow Graphs (MFGs) [9]. MFGs are derived from constraint-based GEMs and an FBA solution for the model. They therefore integrate graph theory with COBRA methods.

MFGs take a cell's biological context into account by incorporating the FBA solution into the graph structure. Hence, MFGs are environment-specific metabolic graphs that account for environmental conditions and genetic perturbations. Figure 2.2 provides an overview of the derivation of MFGs.

### 2.6.1 MFG Algorithm

This section presents how MFGs are computed [9]. The algorithm takes as inputs the stoichiometric matrix $\mathbf{S}$ of a metabolic network, and a pre-computed FBA solution vector $\mathbf{v}^*$.

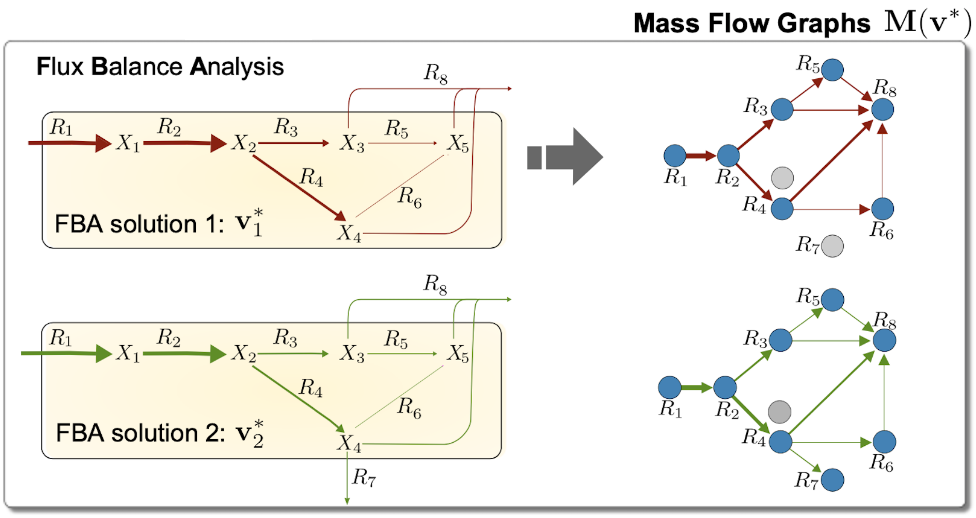
Mass Flow Graphs $\mathbf{M}(\mathbf{v}^*)$
Flux Balance Analysis
FBA solution 1: $\mathbf{v}_1^*$
FBA solution 2: $\mathbf{v}_2^*$
$R_1$ $R_2$ $R_3$ $R_4$ $R_5$ $R_6$ $R_7$ $R_8$
$X_1$ $X_2$ $X_3$ $X_4$ $X_5$

Figure 2.2: Derivation of Mass Flow Graphs from an FBA solution of a metabolic network. Figure adapted from [9].

To compute the MFG, $\mathbf{v}^*$ is unfolded into a vector containing the reaction rates of $2m$ forward and reverse reactions as:

$$\mathbf{v}_{2m}^* = \begin{bmatrix} \mathbf{v}^{*+} \\ \mathbf{v}^{*-} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \text{abs}(\mathbf{v}^*) + \mathbf{v}^* \\ \text{abs}(\mathbf{v}^*) - \mathbf{v}^* \end{bmatrix}. \tag{2.2}$$

The stoichiometric matrix $\mathbf{S}$ is unfolded as:

$$\mathbf{S}_{2m} = [\mathbf{S} - \mathbf{S}] \begin{bmatrix} \mathbf{I}_m & 0 \\ 0 & \text{diag}(\mathbf{r}) \end{bmatrix}, \tag{2.3}$$

which is used to define production and consumption stoichiometric matrices as:

$$\begin{aligned} \mathbf{S}_{2m}^+ &= \frac{1}{2}(\text{abs}(\mathbf{S}_{2m}) + \mathbf{S}_{2m}) \\ \mathbf{S}_{2m}^- &= \frac{1}{2}(\text{abs}(\mathbf{S}_{2m}) - \mathbf{S}_{2m}) \end{aligned}. \tag{2.4}$$

Next, the vector of production and consumption fluxes is computed as:

$$\mathbf{j}(\mathbf{v}^*) = \mathbf{S}_{2m}^+ \mathbf{v}_{2m}^* = \mathbf{S}_{2m}^- \mathbf{v}_{2m}^* \tag{2.5}$$

where $j_i(\mathbf{v}^*)$ is the flux at which metabolite $X_i$ is produced and consumed. $\mathbf{S}_{2m}^+ \mathbf{v}_{2m}^*$ and $\mathbf{S}_{2m}^- \mathbf{v}_{2m}^*$ are equivalent due to the steady state condition, $\dot{\mathbf{x}} = 0$.

Finally, the MFG is defined as:

$$\mathbf{M}(\mathbf{v}^*) = (\mathbf{S}_{2m}^+ \mathbf{V}^*)^\top \mathbf{J}_v^\dagger (\mathbf{S}_{2m}^- \mathbf{V}^*), \tag{2.6}$$

where $\mathbf{V}^* = \text{diag}(\mathbf{v}_{2m}^*)$, $\mathbf{J}_v = \text{diag}(\mathbf{j}(\mathbf{v}^*))$ and $\dagger$ denotes the matrix pseudoinverse.

## 2.7 Graph Theory

Theoretical graph analysis has been applied in Biology to extract information from many kinds of biological networks [33]. This includes measuring node centrality and briefly exploring structural properties of biological networks via clustering.

### 2.7.1 PageRank Centrality

The PageRank algorithm was developed by Brin and Page [34] and used in the search engine of Google for ranking web pages. To estimate the importance of a node in a directed network, it considers the number of incoming edges, the PageRank of the source nodes of these incoming edges, and the number of outgoing links of these source nodes [35].

The PageRank of a node that has incoming edges from nodes $w_1$,$w_2$,...,$w_k$ is computed as:

$$PR(v) = (1-d) + d\left(\frac{PR(w_1)}{C(w_1)} + \cdots + \frac{PR(w_k)}{C(w_k)}\right) \tag{2.7}$$

where $C(w_i)$ is the number of edges leaving node $w_i$ and $d$ is a damping factor between 0 and 1 that is usually set to 0.85 [34].

### 2.7.2 Clustering

The modularity of genome-scale metabolic networks is of particular interest as finding reaction clusters in a network could improve the understanding of the analysed organism. This knowledge could, for example, aid the design of drugs for metabolic disorders [36].

Markov Clustering (MCL) is an unsupervised algorithm for clustering networks based on the simulation of stochastic flows. It is based on the idea that a random walk on a graph which enters a dense cluster will stay in that cluster until it visited many of its nodes [37]. MCL was found to be an effective algorithm for clustering metabolic networks [36].

## 2.8 Node Role Analysis

Node role analysis automatically extracts node roles from large networks [11][12]. Two nodes in a network belong to the same role if they have similar structural behaviour. Figure 2.3a visualises this concept. In this example, we have a red bridge node which connect groups of nodes to the rest of the graph, yellow peripheral nodes which have at most two neighbours and blue nodes which have more neighbours but are not bridges.

In this project, we use and compare two algorithms for node role analysis in directed networks, RolX and Flow Profiles. Both algorithms have been used to extract structural roles from networks in social sciences, ecology and biochemistry [11][12].

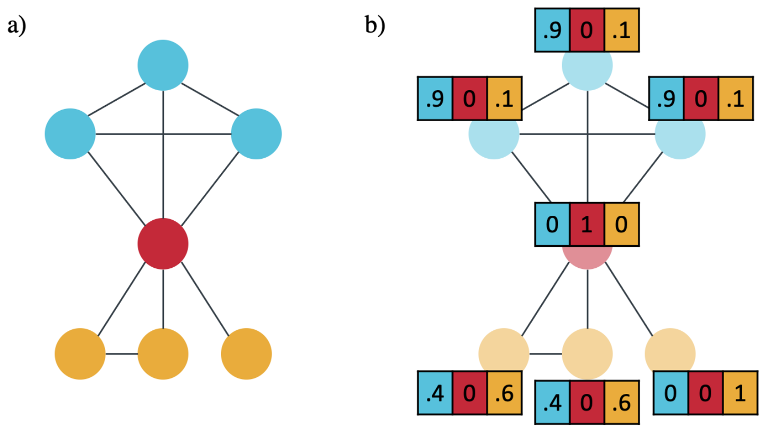

Figure 2.3: Comparison of hard and mixed-membership role assignments. (a) In hard role assignments, nodes are assigned exactly one role. (b) In the mixed-membership approach, nodes have membership in each of the roles. The role memberships capture smaller differences in node structure. Figure adapted from [38].

## 2.8.1 RolX

Role eXtraction (RolX) is an unsupervised approach for automatically extracting structural roles from directed networks proposed by Henderson et al. [11]. It uses a mixed-membership approach which assigns a distribution over the set of discovered roles to each node (see Figure 2.3b).

RolX finds node roles by following three steps: recursive feature extraction, feature grouping and model selection. Based on the publications of Henderson et al., Kaslovsky implemented the recursive feature extraction algorithm and a node role extractor which we used in this project [13].

### 2.8.1.1 Feature Extraction

The Recursive Feature eXtraction algorithm (ReFeX) recursively computes regional features of nodes.

To initialise the algorithm, neighbourhood features are computed which consist of local and egonet features. A node's egonet consists of node itself, all of its neighbours, and all edges within this set of nodes; ReFeX additionally considers the incoming and outgoing edges to the egonet. Local features are measures of node degree. For a weighted, directed graph, they include weighted in-, out- and total degree. Egonet features for nodes of a weighted, directed graph consist of the weighted, directed number of edges within, entering and leaving the egonet.

These initial features are recursively combined into regional features using two operations: means and sums. More specifically, ReFeX computes weighted means and sums of all feature values in a node's egonet, for incoming and outgoing edges separately.

As there is an infinite number of possible recursive features, ReFeX prunes features that do not add any additional information to the set of already computed features. The feature values are mapped to small integers using logarithmic binning. Features are considered duplicate if they are within a fixed range of each other for every node in the network. An undirected graph is constructed with features as nodes that have edges between them if they are duplicate. Then, every connected component in the feature graph is replaced by the node of its feature that was generated with the smallest number of recursions.

The algorithm halts once it reaches an iteration in which no new features are kept after pruning. The features which are in the final pruned graph are used to construct the feature vector of each node. These feature vectors are assembled in the node-feature matrix $V$.

#### 2.8.1.2 Role Extraction

As explained above, the Role eXtraction (RolX) automatically extracts roles from a graph using a mixed-membership approach. Given the number of roles, denoted by $r$, RolX applies non-negative matrix factorisation to create an approximation of the node-feature matrix $V$:

$$V_{n\times f} \approx G_{n\times r} \times F_{r\times f} \tag{2.8}$$

where entries $G_{ij}$ quantify the membership of node $n_i$ in role $r_j$ and entries $F_{jk}$ specify how a membership in role $r_j$ contributes to the value of feature $f_k$. This approximation has rank $r$ equal to the number of roles. Assuming that node roles summarise the behaviour of nodes in the network, these two matrices effectively compress $V$.

#### 2.8.1.3 Model Selection

The role extraction step is carried out for different numbers of roles $r$. In the final model selection step, the resulting models are compared and RolX chooses the best model using minimum description length as performance measure. Minimum description length is defined as the sum of bits required to describe the approximation model and the bits required to specify the approximation error. Hence, RolX favours accurate models with a small number of roles.

### 2.8.2 Flow Profiles

An alternative approach to node role analysis in directed graphs was presented by Cooper et al. who computed node roles via flow profiles of nodes [12].

First, the flow profiles of each node in the network are computed. Second, the similarity between nodes is computed as similarity of their flow profiles. Finally, this similarity matrix is clustered. These clusters correspond to node roles.

A node's flow profile is defined by the overall pattern of incoming and outgoing fluxes. The matrix $X$ of flow profiles is computed from a graph's adjacency matrix $A$ as:

$$X = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} \equiv \left[ \cdots (\beta A^T)^k \mathbf{1} \cdots | \cdots (\beta A)^k \mathbf{1} \cdots \right] \quad \text{for } k = 1, 2, ... \tag{2.9}$$

We have $\beta = \alpha/\lambda_1$, where $\lambda_1$ is the largest eigenvalue of $A$ and $\alpha$ is a scale factor between 0 and 1 that controls the weighting of the local against the global flow structure of the graph. For $\alpha \rightarrow 0$ only paths of length 1 are taken into account, so the flow profiles essentially measure a node's in- and out-degree. For $\alpha \rightarrow 1$, the weight assigned to longer paths increases; the flow profiles take global flows into account.

The similarity of nodes is computed as the cosine distance between their flow profiles. We construct the similarity matrix $Y$ where entry $Y_{ij}$ measures the distance between flow profiles of nodes $i$ and $j$ as:

$$Y_{ij} = \frac{\mathbf{x}_i \mathbf{x}_j^T}{||\mathbf{x}_i|| ||\mathbf{x}_j||}. \tag{2.10}$$

Next, a symmetric graph is created from this similarity matrix. The nodes in the similarity graph are the same as in the original graph. They have edges between them if their flow profiles have non-zero similarity. This matrix can be clustered with any method available for clustering weighted symmetric graphs. Node roles are the resulting clusters and are relatively independent of the chosen clustering algorithm [12].

## 2.9 Machine Learning

In this project, a variety of classification algorithms are used for supervised reaction essentiality prediction.

### 2.9.1 Random Forest

A Random Forest (RF) is an ensemble method; a set of random decision trees is trained to the input data and their outputs are averaged to do predictions [39].

RFs use bootstrapping to generate multiple sets of training samples, a random decision tree is fit to each of these sample sets. Random trees are decision trees which randomly sample a small number of features from the training set and only consider this subset of features to find optimal splits [39]. Random feature selection results in different features used by the different trees, so their errors due to overfitting are hoped to be more independent [39]. To predict new samples, the RF averages over predictions from the set of decision trees [39]. Fernandez-Delgado et al. found RFs to perform best on a variety of datasets [40].

### 2.9.2 Support Vector Machine

A Support Vector Machine (SVM) finds a hyperplane which separates the two classes in the feature space [41]. It aims to maximise the margin that separates the classes which reduces the risk of false classification. SVMs are memory efficient and versatile classifiers which can be used for binary as well as multi-class classification.

### 2.9.3 Logistic Regression

Logistic Regression uses a logistic sigmoid function as transformation to linear output labels to model a binary output variable [42]. The resulting outputs are between 0 and 1 and are interpreted as the probability of a sample to be in class 1. Maximum likelihood estimation is used to find weights that maximise the probability of the data [42].

### 2.9.4 Multilayer Perceptron

A Multilayer Perceptron (MLP) classifier uses a feedforward artificial neural network to learn to map input features to classes [41]. MLPs are universal function approximators and their advantage is the ability to learn non-linear models [41].

However, MLPs require tuning a number of hyperparameters for optimal performance, such as the learning algorithm, the number of layers and the network architecture [43]. Additionally, they are sensitive to feature scaling and to noise in the training data [43].

# Chapter 3

# Implementation

This chapter explains the design and implementation decisions behind MFGpy, a python package for automatically generating and analysing MFGs. This includes our implementation of essentiality and outlier analysis, the two methods for analysing MFGs.

## 3.1 Design Decisions

### 3.1.1 Programming Language

The openCOBRA project developed open-source code for constraint-based analysis of GEMs in MATLAB, Python and Julia [44]. Python is consistently listed as one of the most commonly used programming languages, ranking top 5 in the Stack Overflow Developer Survey 2020 [45]. It comes with a wide array of scientific programming, machine learning, network analysis and visualisation packages. Further, Python is developed under an open-source license, making it freely usable and distributable. Hence, Python was chosen for this project.

### 3.1.2 Object-Oriented Design

COBRApy is openCOBRA's package that provides support for basic COBRA methods in Python [30]. In accordance with the design of COBRApy, we designed MFGpy in an object-oriented fashion. We developed the MFG class to give the user direct access to the MFG, as well as the COBRA model and the FBA solution it was computed from.

### 3.1.3 Data Structures

The adjacency matrices of MFGs are computed following the theoretical algorithm presented in Chapter 2. These matrices are sparse, so we minimise their memory footprint by storing them as sparse matrices. For the MFG of cell line BT-549, this reduces the size required to store the adjacency matrix from 459.1 MB as NumPy array to 48 bytes as SciPy sparse matrix.

In addition, we compute nodes and edges tables of the MFG as pandas DataFrame. The two DataFrames have a total size of 200KB for the BT-549 cell line and provide easy access to a list of active reactions. This is useful for further analysis of the MFGs.

## 3.2 The MFG Class

We created the custom class `MFG` for encapsulating a COBRA model and its MFG. The class also contains methods to analyse, visualise, cluster and export the MFG. An overview of the class structure is displayed in Figure 3.1.

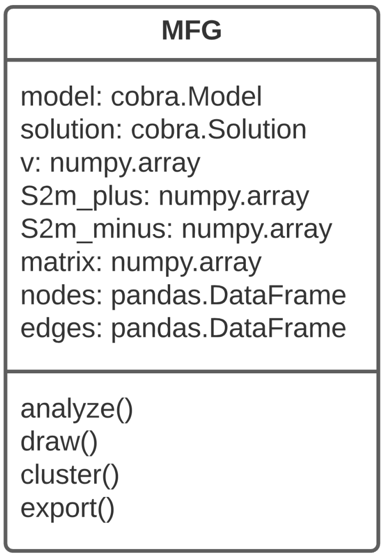

Figure 3.1: UML diagram of the MFG Python class.

### 3.2.1 Creating MFG Objects

To create a new instance of the class, the user calls `MFG` providing it with the `cobra.model` of a metabolic network. Further, they can provide `MFG` with the `cobra.solution` containing the flux vector for generating the graph.

The initialiser is designed to directly compute the adjacency matrix as well as nodes and edges tables of the MFG when a new `MFG` object is instantiated. We thus assign the following attributes during instantiation:

- `model` stores the `cobra.model` corresponding to this MFG
- `solution` is assigned to the pre-computed solution provided by the user if available and is computed using pFBA otherwise
- `v` is the flux vector for computing the MFG that is derived from the solution
- `S2m_plus` and `S2m_minus` are the stoichiometric production and consumption matrices computed from the stoichiometric matrix and the reaction reversibility vector

- `matrix` is the adjacency matrix of the MFG computed from `v`, `S2m_plus`, and `S2m_minus`
- `nodes` and `edges` are the MFG's nodes and edges tables derived from the matrix

The `S2m_plus` and `S2m_minus` matrices are stored to minimise the time needed to generate new MFGs for the same cell. As explained in the next section (Section 3.3), only the reaction bounds are changed for essentiality and outlier analysis. The stoichiometric production and consumption matrices remain constant for all MFGs of the same cell line. Avoiding to re-compute them significantly reduces the time needed for generating a new MFG. Note that, unlike the COBRA Toolbox, COBRApy does not provide access to a reaction reversibility vector. Thus, it has to be computed manually.

### 3.2.2 Visualisation

MFGpy provides the `draw` method for visualising an MFG in Python directly after computing it. For this purpose, we use methods for graph visualisation provided by NetworkX [46]. This enables users to briefly investigate MFGs visually. However, the main purpose of NetworkX is to allow graph analysis rather than perform graph visualisation [46]. We therefore recommend using network visualisation software such as gephi [47] to create high quality visualisations of MFGs, for example for communicating research findings.

### 3.2.3 Clustering

The `cluster` method clusters the reaction-nodes in the MFG with Guy Allard's implementation of MCL (see Chapter 2) [48]. We include this method to facilitate automatic clustering of the MFG instantly after its generation and, if desired, immediate visualisation of the clustered graph.

MCL was used because it effectively extracts clusters from metabolic networks [36]. However, we have not yet evaluated the algorithms effectiveness for clustering MFGs, nor have we investigated suitable values of clustering parameters. The in-depth evaluation of MCL exceeds the scope of this project and is an area of future research.

### 3.2.4 Exporting MFGs

MFGpy provides the `export` method that automates exporting MFGs and storing them for further analysis or visualisation. MFGs can be exported as adjacency matrix in NumPy file format or as nodes and edges tables using Comma Separated Values (CSV).

The sparse adjacency matrix is stored as a NumPy file because of its low memory footprint and fast import and export times. This format is required for further analysis of the MFG in Python, such as node role analysis.

Nodes and edges tables are exported as CSV files so they can conveniently be imported to common visualisation software such as gephi [47].

## 3.3 Analysing Reactions

MFGpy provides the `analyze` method to perform outlier and essentiality analysis of all reaction-nodes in the MFG. We loop over every reaction in the MFG via the list of reaction labels extracted from the nodes table of the original MFG. In every iteration, we knock out reaction *r* using the command `r.knock_out` provided by COBRApy, which sets the reaction's upper and lower bounds to zero. As explained in section 2.4, the bounds of *r* specify the allowed rate at which every metabolite is consumed or produced by *r*.

Outlier and essentiality analysis are performed in parallel, as both require computing FBA solutions for knock-outs of every reaction in the MFG. Hence, computing the two separately would significantly increase the time required for analysis without any additional benefit. However, if users are not interested in outliers, they can decide to only compute reaction essentialities to slightly speed-up essentiality computation. For this purpose, `analyze` has the optional Boolean argument `outliers` which determines whether outliers are computed. It is True by default and can be set to False to only compute essentialities.

### 3.3.1 The Search Space

Each GEM of our cancer cell lines contains 3788 reactions. Running FBA for the knock-out of every one of those reactions is computationally infeasible, but we can reduce this search space significantly.

Recall that the aim of our analyses is to find reactions whose removal from the cell stops or slows cell growth. These reactions have non-zero essentiality; their knock-out reduces the maximal possible growth rate of the cell.

By computing an FBA solution of the metabolic network, we get flux vector *v* which results in maximal cell growth. By the definition of MFGs, all reactions that have non-zero elements in *v* are MFG nodes whereas reactions that have zero elements in *v* are not (see Chapter 2). The latter do not contribute to the optimal solution found with FBA. It follows that only knocking out reactions in the wild type MFG can affect the optimal FBA solution. Hence, we only have to compute the essentiality of reactions in the wild type MFG.

To further reduce the search space, we use parsimonious Flux-Balance Analysis (pFBA) for computing *v* instead of the standard FBA algorithm. As explained in Chapter 2, pFBA is a variant of FBA that minimises the total sum of fluxes whilst optimising the objective function [28].

The significance of this minimisation of the search space can be illustrated using the BT-549 cell line. The original GEM contains 3788 reactions. Running FBA using COBRApy computes a flux vector with 417 non-zero elements. When using pFBA instead, the flux vector only contains 391 non-zero elements. We thus managed to reduce the number of reactions we need to analyse from 3788 to 391, just over 10% of the original set of reactions in the GEM.

### 3.3.2 Essentiality Analysis

We measure the essentiality of reaction $R_j$ as the effect that knock-out of $R_j$ has on the cell's speed of growth. Using COBRA models, reaction knock-out is simulated by setting its upper and lower bounds to zero.

Maximising the biomass components has been identified as a realistic metabolic task for cells of a growing organism [7]. Therefore, a suitable FBA objective function for analysing metabolic networks of cancer cells is maximising the biomass assembly reaction: $Z = F_{BIOMASS}$.

This objective function is used to define the growth rate ratio of reaction $R_j$ as:

$$\Delta Z_{R_j} = \frac{Z_{R_j}}{Z_{WT}} \tag{3.1}$$

where $Z_{R_j}$ is the optimal $Z$ in the cell where reaction $R_j$ was knocked out and $Z_{R_j}$ is the optimal $Z$ in the wild type.

We use the growth rate ratio $\Delta Z_{R_j}$ to define the essentiality of reaction $R_j$ as:

$$e_{R_j} = 1 - \Delta Z_{R_j} \tag{3.2}$$

To compute the essentiality of reactions in a GEM, we have to knock out every reaction in the model individually and perform FBA which is computationally very expensive. Hence, we explore potential solutions to this performance issue in this project.

During analysis, the essentiality values are stored in a list. Once all essentiality values are computed, this list is added as new column to the nodes table.

### 3.3.3 Outliers

As first experimental explorations of the structure of MFGs, we computed node centralities of all nodes in a knock-out MFG and compared it to the wild type MFG.

Figure 3.2 shows the comparison of the wild type MFG of the BT-549 cell line with the MFG resulting from knocking out the Xylulokinase reaction. The plot shows that some reactions undergo atypical changes in PageRank due to the removal of one node from the MFG. These outliers show that graph properties of MFGs capture information beyond FBA.

This discovery stimulated our interest in comparing network centralities of knock-out and wild type MFGs and motivated adding outlier analysis to MFGpy. We generate the MFG of the new network using pFBA solution vector $v_{KO}$. Then, the PageRank percentiles of all nodes in the wild type and the knock-out graph are compared to find outliers. More specifically, a reaction is considered to be an outlier if its PageRank changes by more than a threshold $t$:

$$\Delta PR = |PR_{WT} - PR_{KO}| > t. \tag{3.3}$$

Similar to the essentiality values, the sets of outliers for each reaction knock-out are stored in a list which is added to the nodes table after finishing the analysis. To facilitate an in-depth analysis of the sets of outliers, we store outliers with increased PageRank and outliers with decreased PageRank separately.

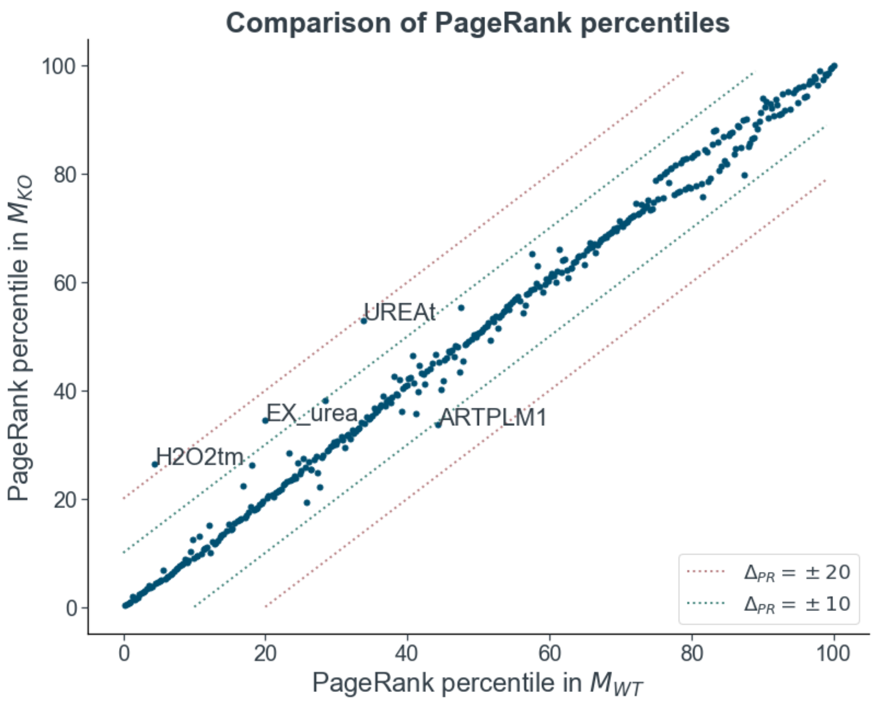

Figure 3.2: Comparison of the PageRank percentiles in the wild type (WT) and Xylulokinase-knockout (KO) MFGs with reactions whose rank changes by more than 10 percentiles labelled.

# Chapter 4

# Prediction of Reaction Essentiality

In this chapter, the machine learning problem for essentiality prediction is presented. I provide an overview of the machine learning pipeline, explain the different feature sets and classifiers used and present how we evaluated their performance.

## 4.1 Overview of the Machine Learning Pipeline

The high-level overview of the machine learning pipeline (see Figure 4.1) explains how the following part of this project integrates with MFGpy.

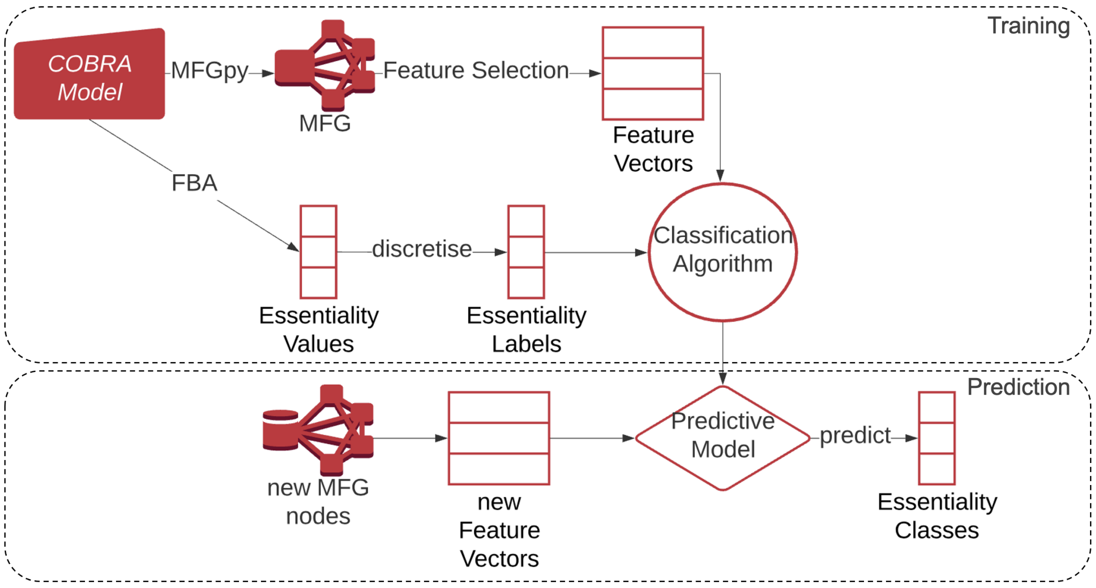

Figure 4.1: The Machine Learning Pipeline for Essentiality Prediction.

The starting point was the COBRA model of a GEM from which we generated data to train predictive models. We used MFGpy to construct the wild type MFG and perform essentiality analysis. This provided FBA essentiality values for all reactions in the MFG.

Essentiality prediction was approached as classification problem, so we converted the FBA essentiality values into essentiality class labels. We investigated both binary and four-class classification.

The adjacency matrix of the wild type MFG was used for feature engineering. We assembled feature matrices for three feature sets – flow profiles, ReFeX features and RolX features. The feature matrices were split into training and test set and used to train a selection of classifiers. Finally, we evaluated the classifiers on the test set and compared their performance. Due to the limited size of the training set we decided to use cross-validation (CV) for choosing hyperparameters; we thus did not have a separate validation set.

## 4.2 Data Collection and Exploration

In order to train a model that could discriminate between essential and non-essential reactions, we had to generate our own training data. We used COBRA models of five cancer cell lines, BT-549, HCT 116, K-562, MCF7 and OVCAR-5 (see Chapter 1). MFGpy was used to compute the wild type MFG of each model and perform essentiality analysis on each MFG. Consequently, we created a dataset of five MFGs containing a total of 1987 reactions and their FBA essentiality. Table 4.1 summarises statistics of the collected data.

| Cell Line | Number of Reaction Nodes | Mean Essentiality |
|---|---|---|
| BT-549 | 391 | $0.266 \pm 0.427$ |
| HCT 116 | 395 | $0.264 \pm 0.425$ |
| K-562 | 420 | $0.249 \pm 0.418$ |
| MCF7 | 395 | $0.264 \pm 0.425$ |
| OVCAR-5 | 386 | $0.269 \pm 0.428$ |
| Total | 1987 | $0.262 \pm 0.425$ |

Table 4.1: Statistics of reactions in the MFGs of five cancer cell lines.

Figure 4.2 shows a rank ordered plot of reaction essentialities in our dataset. We investigated binary classification first as it is a simpler prediction task and the set of reactions naturally divides into two classes. Using threshold 0.5 to divide reactions, we get 1479 essential and 508 non-essential reactions (see Figure 4.3).

To capture mid-essentiality reactions, we extended our labels to the following four classes: *No Change* ($e_{NC} < 0.1$), *Mild Change* ($0.1 \leq e_{MC} < 0.5$), *Severe Change* ($0.5 \leq e_{SC} < 1$) and *Lethal* ($e_{Lethal} = 1$) as shown in Figure 4.2. As shown in Figure 4.3, the resulting class distribution is severely imbalanced; only 109 out of the 1987 reactions in the data set are in classes *Mild Change* and *Severe Change*. The lack of mid-essentiality reactions poses a challenge for training classification algorithms. This issue is further investigated in Section 4.8.

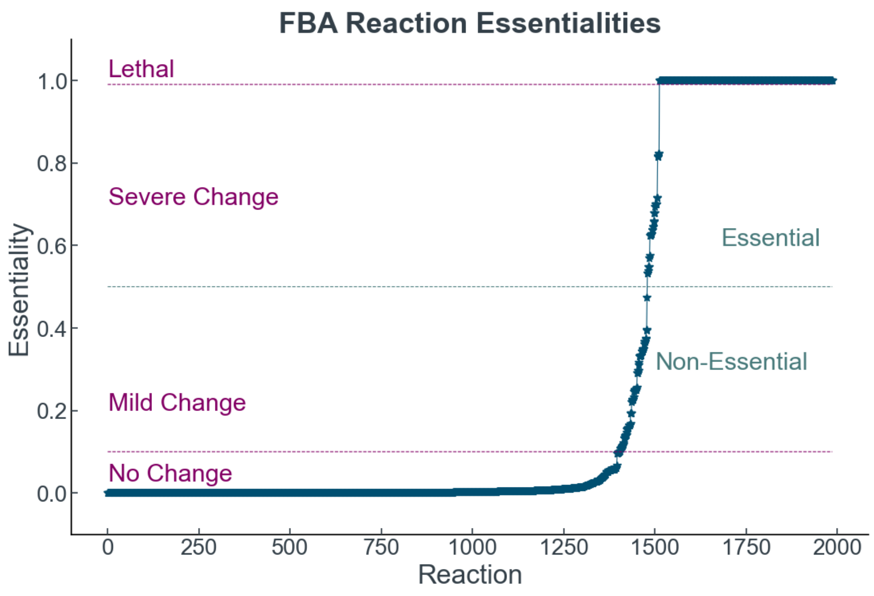

Figure 4.2: Rank ordered plot of reaction essentialities in our dataset with lines indicating the thresholds used for grouping reactions in essentiality classes (red: binary classes, black: four classes)

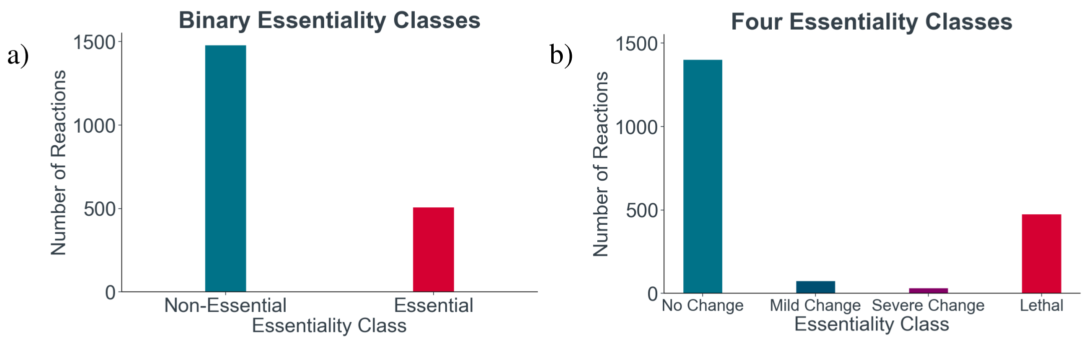

Figure 4.3: The distribution of reaction essentialities when split into (a) two or (b) four classes.

## 4.3 Creating Reaction-Feature Matrices

The next step towards classifying the essentiality of reactions was feature engineering. More specifically, we used MFGs to compute reaction-feature matrices to use as input to classification algorithms.

Figure 4.4 visualises the process of creating reaction-feature matrices from MFGs on the basis of a simple example.

Engineering features fit for a machine learning model requires domain-specific knowledge in metabolic networks and graph theory which made this step especially challenging. There is an unlimited number of possible features that can be derived from graphs, most of which will not provide classification algorithms with information suitable for classifying reaction essentiality.

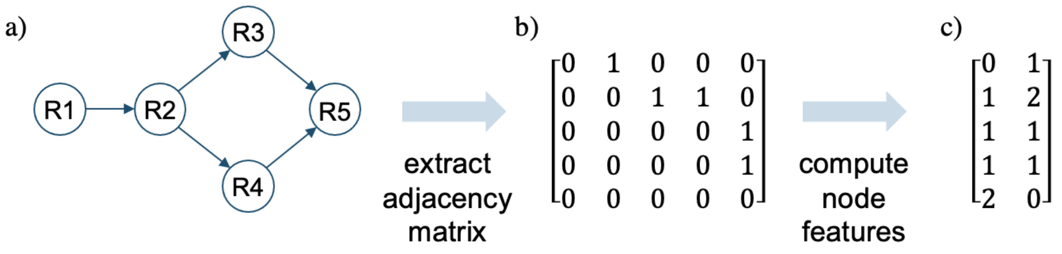
a)
R1
R2
R3
R4
R5
extract adjacency matrix
b)
$\begin{bmatrix}0&1&0&0&0\\0&0&1&1&0\\0&0&0&0&1\\0&0&0&0&1\\0&0&0&0&0\end{bmatrix}$
compute node features
c)
$\begin{bmatrix}0&1\\1&2\\1&1\\1&1\\2&0\end{bmatrix}$

Figure 4.4: An overview of engineering reaction features from MFGs. (a) The MFG is constructed. (b) The adjacency matrix is extracted from the MFG. (c) Structural node features are computed and collected as feature matrix. In this example, a reaction's feature vector contains its in-degree and out-degree in first and second dimension, respectively.

Directed graphs are often analysed via clustering which, however, is not suited for this project. We expect reactions that play a similar role in the cell to exhibit similar structural in the graph, not to be closely connected.

We researched alternative analyses of directed graphs and came across node role analysis. Cooper et al. present this approach as a way to uncover structure in networks where there is an implicit flow transfer in the system [12]. This applies to MFGs as they encode mass flows in the weighted edges between reaction-nodes [9].

Hence, combining MFGs with node role analysis seems very promising for essentiality prediction. We found two approaches to node role analysis in directed networks and decided to explore both in the context of this project.

## 4.3.1 RolX

RolX is an algorithm proposed by Henderson et al.[11][49]. It can be divided into two parts, Recursive Feature eXtraction (ReFeX) and Role eXtraction (RolX), as presented in Chapter 2. We used Kasolvsky's implementation of ReFeX and RolX in Python [13].

Originally, we planned to use the node-role memberships computed with RolX as input features to our classification algorithm. However, we discarded this feature set for various reasons. RolX is an unsupervised algorithm for node-role extraction. We ran it on the five MFGs and noticed that it finds different optimal numbers of roles for different MFGs. We can fix the number of roles, but this still does not enable combining the feature matrices from multiple MFGs because each role is not clearly defined and was computed based on a different set of features extracted by ReFeX.

Henderson et al. used RolX for transfer learning. They trained it on one graph and applied the extracted features and node roles for classification on a new, unknown graph. We could have followed this approach but decided against it as it only allows using one graph for training the algorithm. This would reduce our training dataset from nearly 2000 nodes to no more than 420 nodes and we would then only have access to 22 mid-essentiality reactions at training time. Furthermore, the variation in RolX roles across cell lines indicates that they are not a reliable measure for analysing MFGs. Hence, we did not pursue this idea further.

### 4.3.2 ReFeX

As explained above, RolX features were rejected as they cannot easily be combined across MFGs. Note that the extracted roles by RolX are a low-rank approximation of the input features. Hence, we explored the suitability of the ReFeX features instead of using node roles.

ReFeX automatically extracts a variety of structural features for each node. The recursive computation of the flow structure of each node and its neighbours yields a promising combination of local and global structural features.

Table 4.2 shows an overview of the features ReFeX extracted from the five MFGs. Similar to RolX, ReFeX is unsupervised and consequently does not extract the same structural features for different MFGs.

| Cell Line | Extracted Features | Common Features | % kept |
|---|---|---|---|
| BT-549 | 31 | 26 | 83.9 |
| HCT-116 | 34 | 26 | 76.5 |
| K-562 | 31 | 26 | 83.9 |
| MCF7 | 34 | 26 | 76.5 |
| OVCAR-5 | 30 | 26 | 86.7 |

Table 4.2: The features automatically extracted by ReFeX. The features which are in all feature sets and thus kept vs the features that were discarded.

We found that 26 of the extracted features were in the feature set of all cell lines.If we keep these 26 features for every graph, we use 81.5 % of the extracted features. This enables us to combine all five MFG feature matrices into one 26-dimensional feature matrix which contains the majority of information from the extracted ReFeX features. Since these features were extracted for every MFG, they are likely to be useful for MFGs of similar cells which the classifier could be used on in the future.

### 4.3.3 Flow Profiles

We explored flow profiles, a concept developed by Cooper et al. [12], as alternative feature set. For this purpose, we implemented an algorithm for computing and clustering flow profiles following the theory presented in their paper (see Chapter 2).

The core of the implementation is the computation of a matrix $X$ of flow profiles from the adjacency matrix of the MFG. Then, similarity matrix $Y$ is computed as pairwise cosine similarity between the flow profiles in $X$.

Node roles were then computed by clustering the similarity matrix $Y$. According to Cooper et al.[12], the specific clustering algorithm does not have a strong influence on the resulting clustering of $Y$. The authors chose a variant of spectral clustering for this purpose and as they seemed to achieve good results, we followed their approach.

We first investigated reaction essentiality in the clusters extracted from $Y$. However, the algorithm did not automatically separate essential from non-essential reactions. Therefore, we used the flow profiles in $X$ as input features for classification.

#### 4.3.3.1 Hyperparameters of the Algorithm

The algorithm has two hyperparameters, $k_{max}$ and $\alpha$. It computes the number of incoming and outgoing paths for each node up to length $k_{max}$. We set $k_{max}$ equal to the MFG's directed diameter as it equals the longest shortest path in the graph. This extracts as much information from the MFG as possible. All five MFGs had a directed diameter of length 10, so constant $k_{max}$=10 was used for computing flow profiles. We then combined the five feature matrices which yielded one final 20-dimensional feature matrix containing all 1987 samples.

The second hyperparameter is the scale factor $0 < \alpha \leq 1$, which controls the weight assigned to local relative to global graph structure. Multiplying the adjacency matrix by $1/\alpha$ ensures convergence of the column values to $\lim_{k\to\infty} \frac{||A^{k+1}||}{||A^k||} \to \lambda_1$. For small $\alpha$, nodes are classified based on local properties and as $\alpha$ grows towards 1, longer paths gain influence. To find the optimal value of $\alpha$ for essentiality prediction, we performed an exhaustive search over $\alpha = 0.1, 0.2, ..., 1$.

## 4.4 Classification Algorithms

We trained four different classification algorithms and compared their performance. The four algorithms we used were Random Forests, Support Vector Machines, Logistic Regression and Multilayer Perceptrons (explained in Chapter 2).

We used sklearn's implementations as the nature of this project is exploratory. Before developing complicated classification algorithms, we had to determine whether the available feature sets are suited for predicting reaction essentiality.

### 4.4.1 Evaluation of Classifiers

Crucial for classification models is their ability to make accurate predictions and to generalise well. We thus needed to evaluate and compare the different classification models we trained.

After feature engineering, we obtained two feature matrices of our dataset. As the total size of the dataset was small (1987 reactions), it was decided to use stratified 5-fold cross validation to choose hyperparameters and compare models and features at training time.

Cross validation removes the need of creating a separate validation set. Hence, we conducted experiments using a stratified train-test split (80%, 20%).

To evaluate the performance of our essentiality prediction algorithms, the following metrics were tracked:

1. Confusion Matrix
2. Precision, Recall and F1-Score
3. Overall Prediction Accuracy and Accuracy by Class

The confusion matrix allowed a detailed evaluation of the classifier's performance on each essentiality class, both for binary and four-class classification.

The precision, recall and F1-score (for each class) enable understanding how well the classifier performs on the two classes and quantifies this performance. This enables comparing the different models and feature sets and choosing hyperparameters. In addition, for the binary classifiers we used precision-recall curves to visualise this measure.

Finally, we analysed the overall and class-specific accuracy. The overall accuracy is the simplest quantity to summarise the performance of a classifier and to compare different classifiers and features for the same dataset.

## 4.5 Baseline Models

We began our classification experiments focusing on binary classification. We used both feature sets, ReFeX features and flow profiles (for $\alpha = 0.5$), as input for the four classification algorithms. For training the first models, we used sklearns's standard hyperparameters of each classification algorithm.

Prior to training the first models, we used stratified sampling to set aside 20% of the input samples as test set. These 398 samples were not used in training. Instead, we used stratified 5-fold CV for evaluating the different models and feature sets. Figure 4.5 shows the averaged accuracy of each classification algorithm for the five CV folds.

The best classifier for both feature sets was the Random Forest classifier which had an average CV accuracy of 88.6% (ReFeX) and 86.6% (Flow Profiles). SVM, Logistic Regression and MLP classifiers only achieved accuracies of 72.6% to 76.1% on either feature set. As presented in Section 4.2, 74.4% of training samples are *Non-Essential*. Hence, the SVM, Logistic Regression and MLP performed similar to a classifier that assigns all data points to the larger class.

Noticeable is the particularly large variation of prediction accuracy of the MLP across folds, as shown by the standard deviation plotted as error bars in Figure 4.5.

To gain a deeper insight into the classifiers' performance, we investigated the confusion matrices for each classifier. The cumulative confusion matrices showed that SVM, Logistic Regression and MLP classify nearly all reactions as *Non Essential*; they hardly learn anything from the data.

Our baseline model for further evaluations is the Random Forest classifier on ReFeX features, which achieved a prediction accuracy of 88.6%.

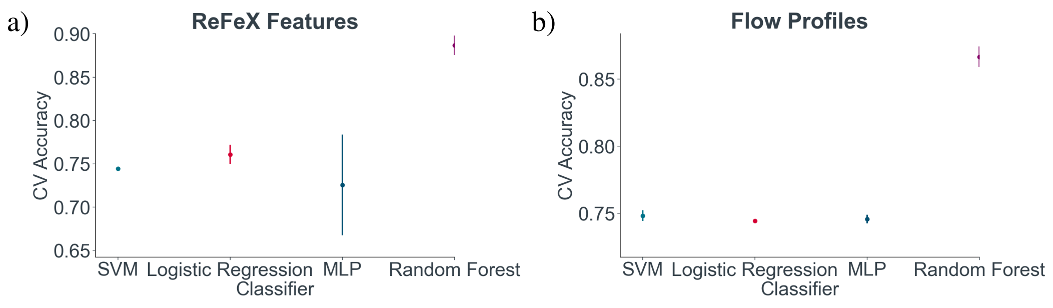

Figure 4.5: 5-fold CV accuracy of each classifier on (a) ReFeX features and (b) Flow Profiles.

## 4.6 Feature Normalisation

Classification models, especially distance-based classifiers such as Logistic Regression and SVM, are impacted by differently scaled dimensions in the data.

We investigated the distribution of feature values for every feature dimension. Figure 4.6 shows the results of this analysis on ReFeX feature. The different dimensions have very different scales which arises from how they were computed; as explained in chapter 2 ReFeX recursively computes means and sums over feature values of a node's neighbours.

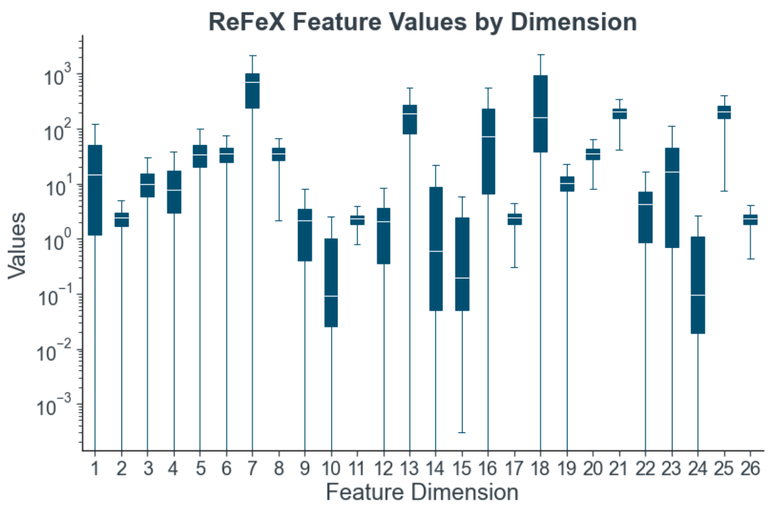

Figure 4.6: Distribution of feature values for each dimension in raw ReFeX features. The y-axis is plotted in log-scale because of the large difference in size across feature dimensions.

We normalised both feature sets and explored how this affects classification performance of the different models. Normalisation was performed by subtracting the mean of every feature dimension of the training samples from the values in every feature dimension and dividing by the standard deviation of that dimension:

$$X_{normalised} = \frac{X - \mu}{\sigma} \tag{4.1}$$

Re-training the classification algorithms on normalised features improved the performance of all four classification algorithms as shown in Figure 4.7. The most significant improvement was observed for the MLP classifier. Its CV accuracy improved from 72.6% to 83.6%. Based on these results, we limited further investigations to the Random Forest classifier and the Multi-Layer Perceptron only. Extensive hyperparameter-tuning could improve the accuracy of the SVM and the Logistic Regression classifiers but we did not explore this further.

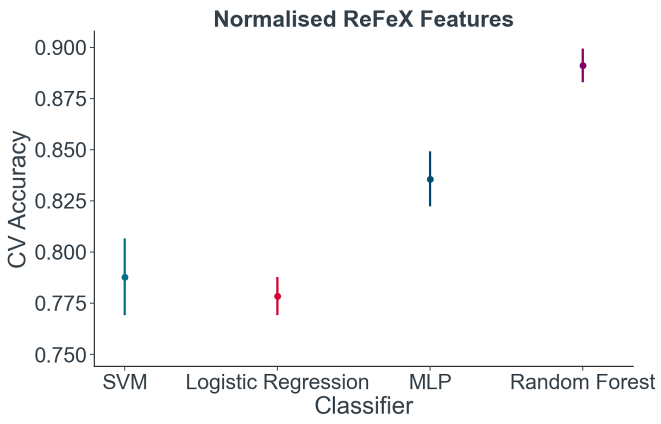

Figure 4.7: Average Cross-Validation (CV) accuracies on normalised ReFeX features.

The same analysis was conducted on the Flow Profile features. However, as explained in Section 4.3.3, the computed column values converge to the largest eigenvalue of the MFG's adjacency matrix. As they are already in a similar range, normalising features did not improve classification performance and was thus omitted.

## 4.7 Hyperparameter Tuning

### 4.7.1 Flow Profile Scale Factor $\alpha$

As explained in Section 4.3.3, the algorithm for computing node flow profiles can vary the weighting of the local vs the global flow structure. Cooper et al. [12] found that values of $\alpha \to 1$ resulted in robust clusterings of the similarity matrix.

Instead of clustering the flow profile similarity matrix, we use the flow profiles as input to classification algorithms. Hence, we decided to investigate the effect of varying values of alpha on classification performance. As shown in Figure 4.8, the Random Forest classifier achieves very similar classification accuracies for all values of $\alpha$. Only the CV accuracy for $\alpha = 0.1$ is circa 2% lower than the rest.

We performed the same analysis using the MLP classifier. However, independent of the value of alpha, its classification performance remained between 74.4% and 75.4%. The detailed results were thus omitted.

As a result of this analysis, we picked $\alpha = 0.8$ for further computations. However, as stated above, classifier performance was found to be fairly independent of the chosen value of alpha.

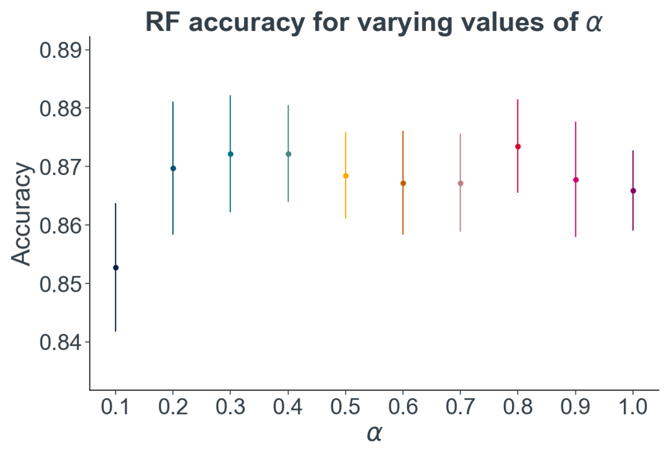

Figure 4.8: Cross-validation accuacy of Random Forests for different values of the Flow Profile hyperparameter $\alpha$.

## 4.8 Multi-Class Essentiality Prediction

Once we had thoroughly explored binary essentiality prediction, we returned to the four-class classification problem. As we are still working with the same dataset our findings from binary essentiality prediction strongly influenced our approach to multi-class prediction. More specifically, we decided to only explore the suitability of Random Forest and MLP classifiers. They delivered significantly better results in binary prediction than the Logistic Regression and the SVM classifier (see Section 4.6). Furthermore, both models can easily be used for multi-class prediction which makes them very suitable for this study.

### 4.8.1 Multi-Class Baseline

Since Flow Profiles and ReFeX features are part of two alternative approaches to node role analysis, we continued examining both datasets for the multi-class problem although we had achieved better results using (normalised) ReFeX features.

Figure 4.9 presents our first experiments with four-class essentiality prediction. The Random Forest classifier achieved a good overall classification accuracy of 86.2% on normalised ReFeX features and 81.2% on Flow Profiles but struggled to classify mid-essentiality (*Mild Change* and *Severe Change*) reactions correctly as shown by the confusion matrices.

The MLP classifier had a lower overall accuracy of 80.5% on ReFeX features. However, it had a slightly better classification accuracy on mid-essentiality reactions. As shown in Figure 4.9, it struggled to learn from Flow Profiles. It classified nearly all reactions as *No Change* resulting in an overall CV accuracy of 70.5 %.

### 4.8.2 Hyperparameter Tuning

Motivated by the MLP's high classification accuracy on the two mid-essentiality classes, we decided to explore if tuning the model's hyperparameters can improve its classifi-

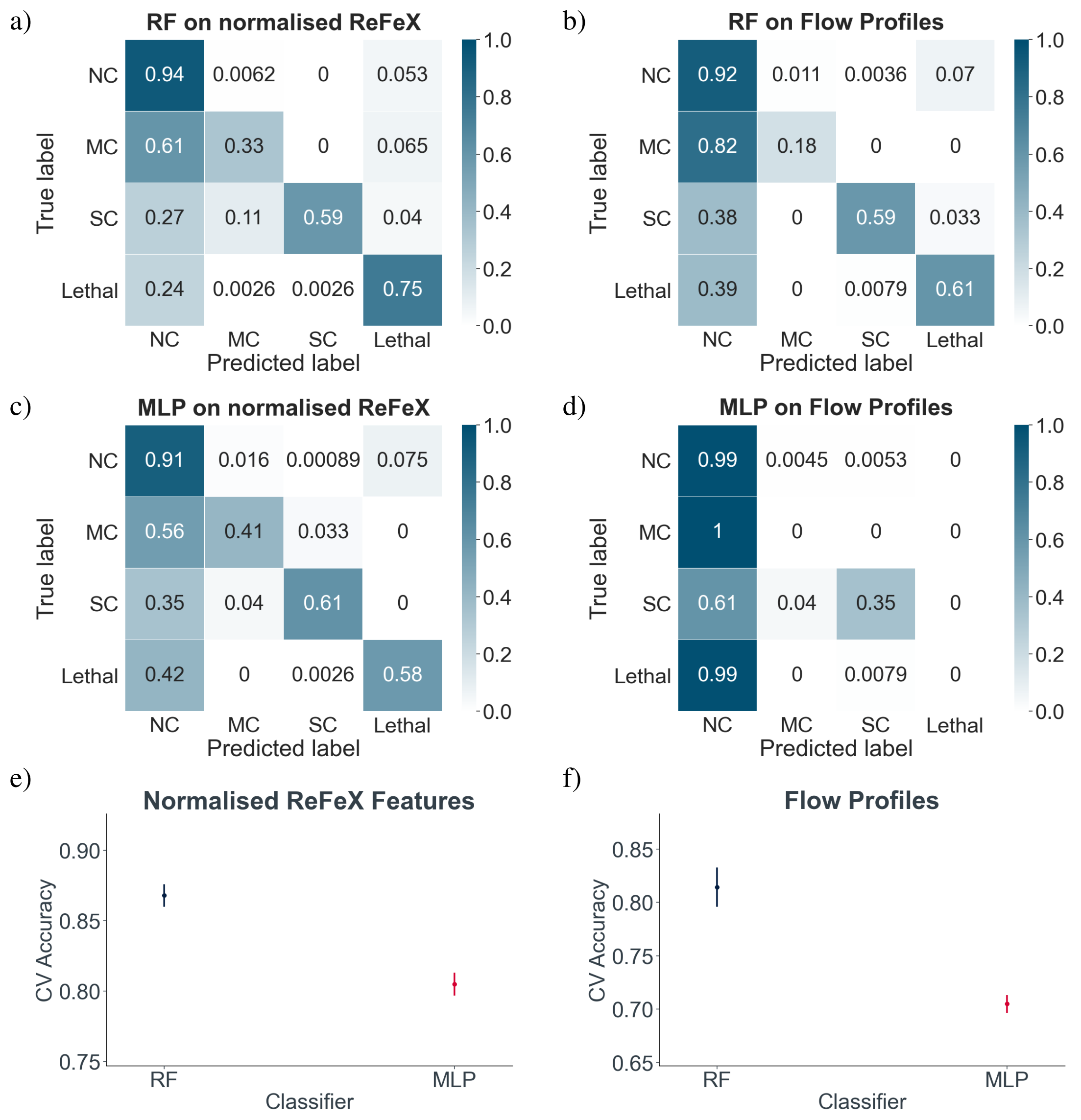

Figure 4.9: First experiments on predicting the four essentiality classes *No Change* (NC), *Mild Change* (MC), *Severe Change* (SC) and *Lethal*. Confusion matrices of the Random Forest (RF) classifier on (a) normalised ReFeX features and (b) Flow Profiles. Confusion matrices of the MLP classifier on (c) normalised ReFeX features and (d) Flow Profiles. A comparison of the classifiers' average Cross-Validation (CV) accuracy on (e) normalised ReFeX features and (f) Flow Profiles.

cation performance.

A variety of hyperparameters were tested for the MLP by performing an exhaustive search over different hyperparameter configurations. Furthermore, we increased the maximum number of iterations to 800. Previously, it was set to 200 which is sklearn's standard value but this halted the algorithm before the optimisation reached convergence. Again, stratified 5-fold CV was used to compare the different models using the training set only. The hyperparameters tested and their configuration which produced the highest accuracy for the MLP are shown in Table 4.3.

| Hyperparameter | Optimal Value |
|---|---|
| Sizes of Hidden Layers | (50,100,50) |
| Hidden Layer Activation Function | Tanh |
| Optimisation Algorithm | Adam |
| Initial Learning Rate | 0.01 |
| Learning Rate Schedule | Constant |
| L2-Regularisation Penalty | 0.005 |

Table 4.3: The tested MLP hyperparameters and their best configuration.

Figure 4.10 shows the confusion matrix of the resulting MLP classifier. Hyperparameter tuning improved the MLP's overall accuracy from 80.5% to 86.8%. This is slightly higher than the previously best model, the Random Forest classifier trained on the same feature set. It had an overall accuracy of 86.2%.

Hyperparameter tuning improved the MLP's predictions across all four classes. Especially good is its recall on mid-essentiality classes. It outperforms all previous classifiers by classifying 60% of *Mild Change* and 89% of *Severe Change* reactions correctly.

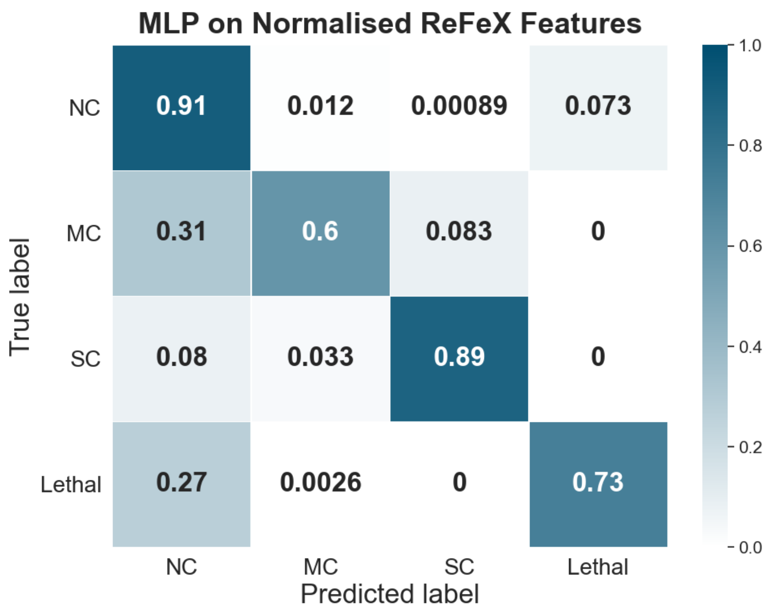

Figure 4.10: Average cross-validation confusion matrix of the MLP classifier predicting the four essentiality classes *No Change* (NC), *Mild Change* (MC), *Severe Change* (SC) and *Lethal* after hyperparameter tuning.

# Chapter 5

# Results

In this chapter, I present the results of this project. First, I present the results from FBA essentiality and outlier analysis of the five cell lines. Second, I summarise our findings from node role analysis. Finally, I present and evaluate the best essentiality prediction model.

## 5.1 Essentiality Analysis

Essentiality analysis strongly influenced the development of the machine learning algorithms. Hence, most findings were already presented in Section 4.2 and I provide only a brief summary here.

We computed and analysed the FBA essentiality of reactions in the five cell lines BT-549, HCT-116, K-562, MCF7 and OVCAR-5. The results are presented in Figure 5.1. It shows that the number of reactions in each essentiality class is nearly constant across the five MFGs. On average, each MFG contains 95 *Lethal*, 7 *Severe Change*, 15 *Mild Change* and 280 *No Change* reactions.

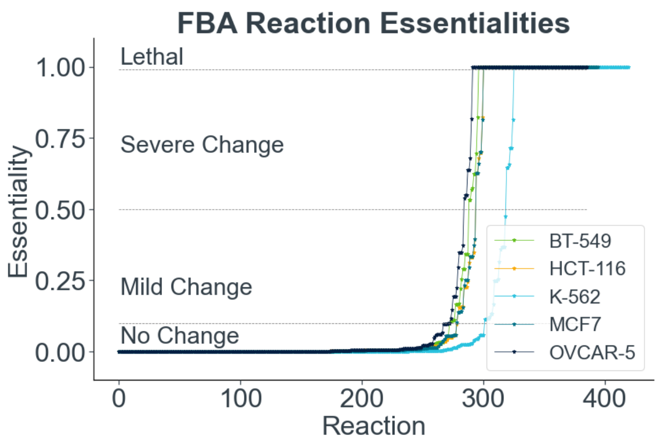

Figure 5.1: FBA essentiality of all reactions in the MFGs of the BT-549, HCT-116, K-562, MCF7 and OVCAR-5 cell lines. All other reactions in each GEM are inactive; they are in the *No Change* class.

## 5.2 Outlier Analysis

We computed the outliers of every reaction knock-out for varying PageRank percentile thresholds. We found that only a limited set of reactions appears as outliers and that certain groups of reactions commonly appear together. Figure 5.2 shows positive and negative outliers of the five cell lines for threshold 20.

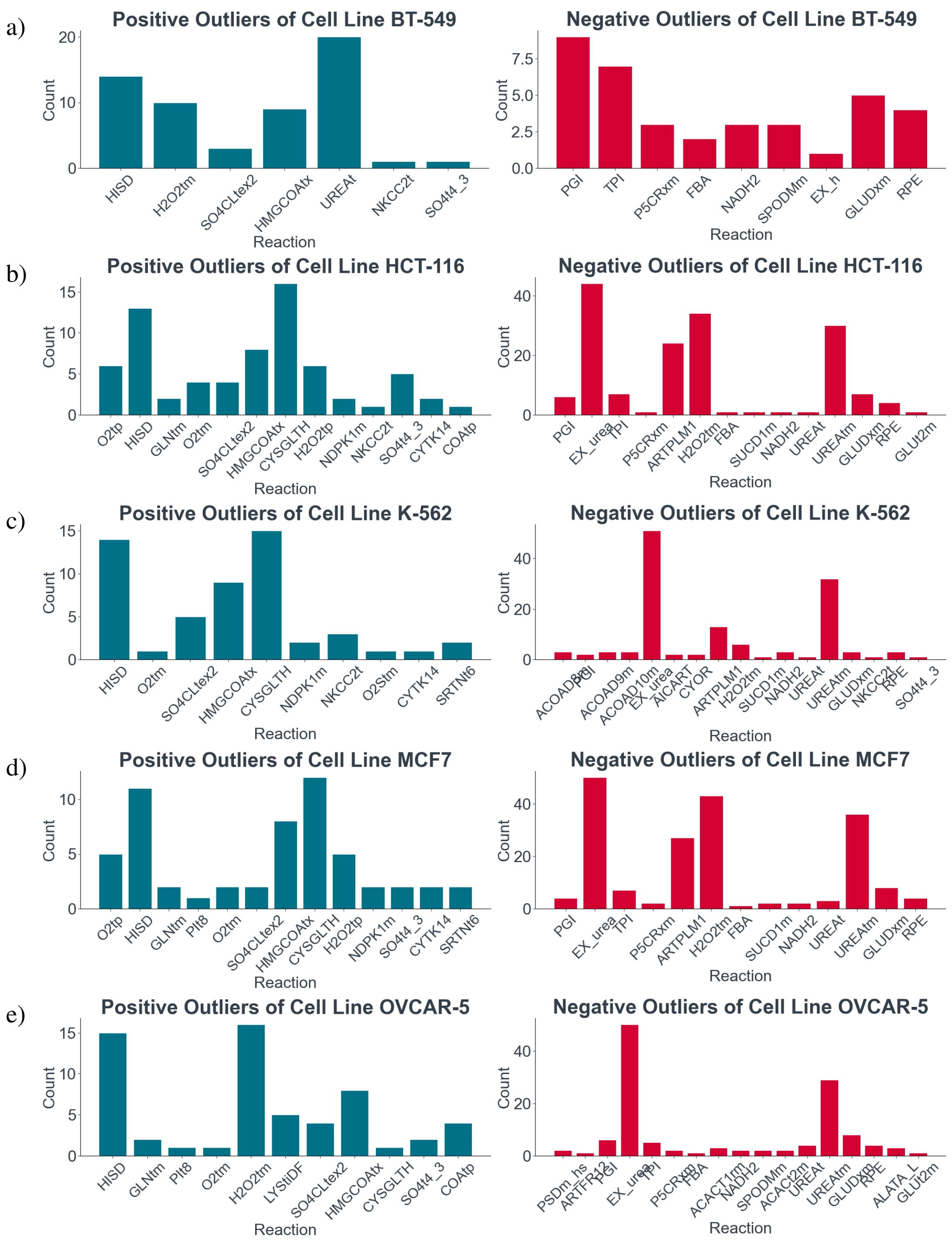

Figure 5.2: Histograms of positive and negative outliers of single reaction knock-outs in cancer cell lines (a) BT-549, (b) HCT-116, (c) K-562, (d) MCF7 and (e) OVCAR-5.

## 5.3 Binary Essentiality Prediction

As presented in Chapter 4, a Random Forest classifier on the normalised ReFeX feature set performed best at binary essentiality classification. To estimate the model's generalisation performance, we trained it on the complete training set and evaluated its performance on the test set.

To accurately simulate the classifier's performance on unknown MFGs, we normalised the features in the test set by applying the following formula for each dimension D:

$$X_{test,normalised} = \frac{X_{test} - \mu_{train}}{\sigma_{train}} \tag{5.1}$$

where $\mu_{train}$ and $\sigma_{train}$ are the vectors of mean and standard deviation per dimension derived from the training set.

Figure 5.3 visualises the classifier's performance on the test set. The final model had a training accuracy of 99.9% and a test accuracy of 89.9%. The confusion matrix on the test set shows, that it correctly classified 95% of *Non-Essential* reactions and 79% of *Essential* reactions. The precision-recall curve visualises that this classifier reaches a recall of about 20 % without predicting any *Non-Essential* reactions to be *Essential* and maintains a recall of about 90% up to a precision of over 80 %. Our model separates the two classes well but not perfectly.

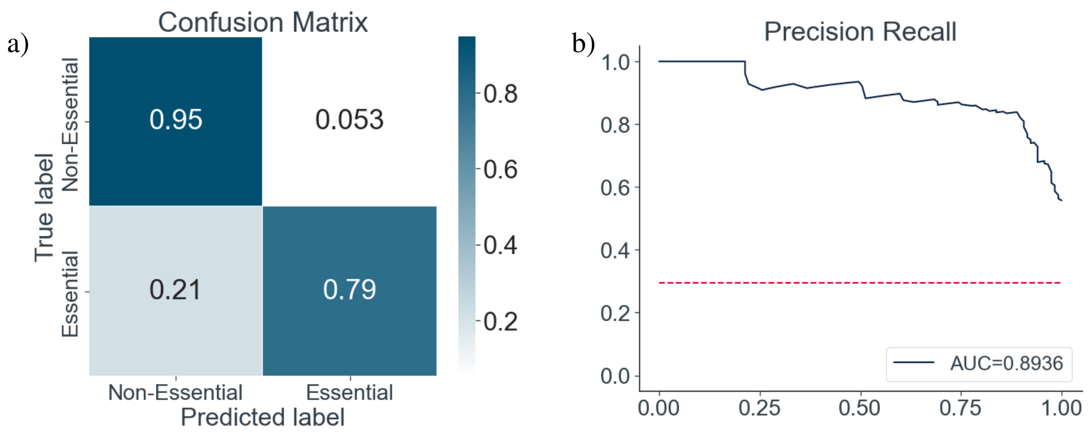

Figure 5.3: Test performance of the binary Random Forest classifier, evaluated using (a) the confusion matrix and (b) the precision-recall curve.

The detailed classification report in Table 5.1 shows that the model achieves similar precision for both essentiality classes. However, recall and F-Score for *Essential* reactions is notably worse than for *Non-Essential* reactions. The class average is the unweighted average of precision, recall and F-Score in both classes. The weighted average is the average of these values weighted by the number of samples in each class. For this classifier, the class average is 1 - 3% lower than the weighted average; the classifier performed slightly better on *Non-Essential* Reactions than on *Essential* reactions.

**Random Forest (Binary Classification)**

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Non Essential | 0.91 | 0.95 | 0.93 | 281 |
| Essential | 0.86 | 0.79 | 0.82 | 117 |
| Class Average | 0.89 | 0.87 | 0.88 | 398 |
| Weighted Average | 0.90 | 0.90 | 0.90 | 398 |

Table 5.1: Classification report of Random Forest classifier on the test set.

A possible explanation is that only 30% of training samples are *Essential* reactions, hence the classifier is biased towards classifying reactions as *Non-Essential*.

## 5.4 Four-Class Essentiality Prediction

The MLP classifier and the Random Forest classifier had similar training performance. We thus evaluated both models on our test set. Again, we trained the final models on the complete training set and evaluated them on the test set. Confusion matrices for both classifiers are presented in Figure 5.4.

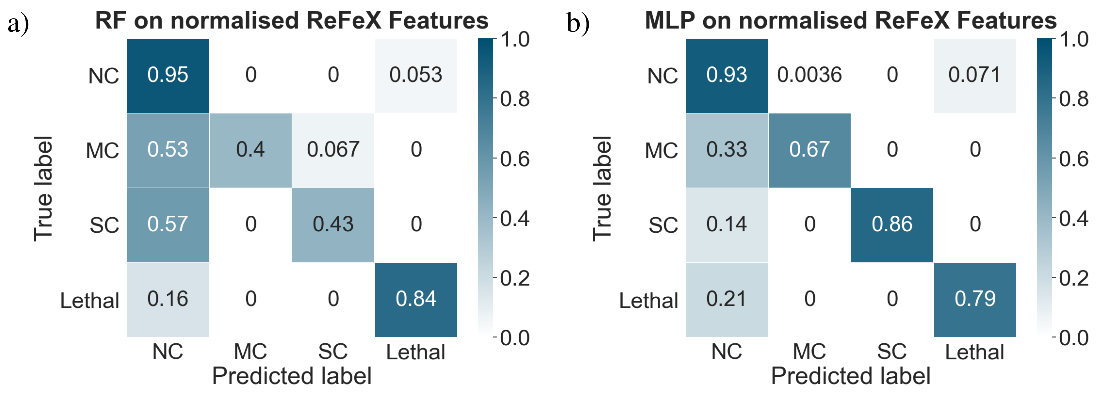

Figure 5.4: Confusion matrices of the final (a) Random Forest (RF) and (b) MLP classifier evaluated on the test set.

The Random Forest classified 89.1% of test samples correctly. The detailed report of its test performance is shown in Table 5.2.

It performs best on *No Change* reactions, with an F-Score of 93%, as well as on *Lethal* reactions where it has an F-Score of 84%. However, the F-Score on the two mid-essentiality classes, *Mild Change* and *Severe Change*, is only 40 - 43 %. Comparing the class and weighted averages, we see that the class-specific recall is 24% lower and the class-specific F-Score is 19% lower than the weighted results. The classifier struggled to predict intermediate classes.

One reason is the very limited availability of reactions with intermediate essentiality. *Mild Change* and *Severe Change* reactions together make up only 5.5% of the training data. We could explore if generating more training data from additional cell lines improves classification performance, especially if researchers are interested in mid-essentiality reactions.

**Random Forest**

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|
| No Change | 0.91 | 0.95 | 0.93 | 281 |
| Mild Change | 1.00 | 0.40 | 0.57 | 15 |
| Severe Change | 0.75 | 0.43 | 0.55 | 7 |
| Lethal | 0.84 | 0.84 | 0.84 | 95 |
| Class Average | 0.87 | 0.65 | 0.72 | 398 |
| Weighted Average | 0.89 | 0.89 | 0.89 | 398 |

Table 5.2: Multi-class classification report of the Random Forest classifier on the test set.

The MLP classifier had an overall classification accuracy of 88.1% on the test set. In contrast to the Random Forest, its average performance by class is more similar to the weighted average performance.

**Multi Layer Perceptron**

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|
| No Change | 0.91 | 0.93 | 0.92 | 281 |
| Mild Change | 0.91 | 0.67 | 0.77 | 15 |
| Severe Change | 1.00 | 0.86 | 0.92 | 7 |
| Lethal | 0.79 | 0.79 | 0.79 | 95 |
| Class Average | 0.90 | 0.81 | 0.85 | 398 |
| Weighted Average | 0.88 | 0.88 | 0.88 | 398 |

Table 5.3: Multi-class classification report of the MLP classifier on the test set.

# Chapter 6

# Conclusion & Future Work

In this report, we presented MFGpy, a software package which facilitates the generation and mathematical analysis of cell line specific MFGs from COBRA models. We used MFGpy to construct and analyse the MFGs of five cancer cell lines. Finally, we presented classification models which successfully predicted FBA essentialities of reactions in MFGs based on network properties.

## 6.1 Results

We developed MFGpy, a software package implementing the novel MFG algorithm developed by Beguerisse et al. [9]. MFGpy uses COBRApy methods for constraint-based modelling. MFGpy was used to generate MFGs from the COBRA models of the five cancer cell lines BT-549, HCT-116, K-562, MCF7 and OVCAR-5 and to analyse reaction essentialities in each network. We found that each cell contains, on average, 16 reactions causing a mild reduction in growth rate, 5 reactions causing a severe reduction and 97 lethal reactions.

In the second part, we created three sets of structural features on reaction nodes from the MFGs. We used the feature sets and an array of classification algorithms to predict the FBA essentiality of reactions. This was approached as binary and four-class classification problem. The best predictive models were Random Forest classifiers which achieved an overall accuracy of 89.9% in the binary problem, and 89.1% in the four-class problem. The limited number of reactions with mid-range essentiality caused low prediction performance of the four-class model on *Mild Change* and *Severe Change* reactions.

## 6.2 Answering the Research Question

As stated in Chapter 1, the aim of this project was to explore whether machine learning algorithms can predict the biological essentiality of metabolic reactions based on the properties of MFGs.

Our initial idea was to use automatically extracted node roles from MFGs. For various reasons, we found node roles unsuited for this task and rejected this idea.

Instead, we used the input features to role extraction algorithms for essentiality prediction. We evaluated a variety of machine learning algorithms using these structural features. The best model was a Random Forest classifier on normalised ReFeX features which could discriminate essential from non-essential reactions with a test accuracy of 89.9%. Hence, machine learning algorithms can use structural properties of MFGs to accurately predict reaction essentiality.

## 6.3 Strengths and Weaknesses of the Method

In this project, we presented a novel approach to the prediction of reaction essentialities in GEMs. We developed MFGpy which automates the generation of MFGs from COBRA models in Python. MFGpy relies on open-source software only and will be openly available on github. This agrees with the purpose of the openCOBRA project which is to promote constraint-based research through the distribution of freely available software. Following the design of COBRApy, we developed MFGpy in an object-oriented fashion enabling researchers to easily combine both packages for their individual analyses.

We built a number of MFGs and used FBA to compute the essentiality of every reaction node in the graphs. We then applied node role analysis to extract structural features on reaction nodes from the MFGs. Together with manually computed FBA essentiality values, we used the feature matrices to train classification algorithms which successfully predict reaction essentiality classes. To the best of the author's knowledge, machine learning has not previously been applied to reaction essentiality prediction in cancer cells.

The classifiers we trained can predict the essentiality of reactions in new MFGs. Hence, reaction essentiality can now be studied from the wild type MFG of a cell's metabolic network only. This requires just one execution of FBA, namely, to generate the MFG, instead of roughly 400, as before.

A notable strength of our methodology was that reproducible processes were prioritised. We used 5-fold Cross-Validation for tracking performance during training time and kept 20% of the dataset completely unseen until the very final model evaluation. The performance metrics of the final models on the test set thus offer an accurate estimate of the model's generalisation performance.

The biggest weakness of this project was that, due to limited time and resources, the training data for the essentiality prediction models was collected from five cancer cell lines only. This especially affected system performance on predicting mid-essentiality reactions. In addition, the five cell lines are fairly similar as they all originate from human cancer tissue. A robust data collection process would have included metabolic networks from a more diverse range of cells. However, as we focused on cancer research as the main area of application, the accuracy of essentiality predictions on metabolic networks of other cells is of less interest.

Another important weakness is that our reaction essentiality values originate from computational simulations using FBA. Accurately simulating the metabolic networks of human cells is especially challenging due to their high complexity [50]. Hence, the predictions produced by FBA must be verified. Nonetheless, various studies found that FBA predictions of growth rate and gene essentiality agree well with experimental measurements [27].

## 6.4 Further Study

We answered the research question and fulfilled the main objective of this project. Nevertheless, there are a lot of interesting directions for future work.

In our analysis of outliers, we discovered certain groups of reactions which often appear in outlier sets together. These groups of reactions should now be further analysed. For example, using flux enrichment analysis which finds significant metabolic pathways from a set of reactions.

As mentioned in the analysis of method weaknesses, our machine learning models were trained using a very limited amount of data. For further study, additional training data should be generated, for example from GEMs of different cancer cell lines constructed by Yizhak et al. [4]. We expect that increasing the size of the training dataset will improve the accuracy of essentiality predictions especially for mid-essentiality reactions.

The biological accuracy of predictions from our machine learning algorithms has to be thoroughly evaluated. Firstly, available scientific literature and metabolic databases should be searched for data on reaction essentiality in the cells we analysed. Secondly, the predictive models should predict the essentiality of reactions in other cells where (FBA) reaction essentiality values are available to evaluate their performance on unknown cells. Finally, if further information is required to evaluate model predictions, biological experiments can be executed to investigate reaction essentiality in specific cells under different conditions.

### 6.4.1 Plan for MInf Part 2

In this project, we have shown that reaction essentiality can be predicted based on structural features of MFGs. Compared to flux balance analysis predictions in cancer cells, our classifiers achieved over 88% accuracy. In the second part of this project, we will extend this analysis in two areas.

We will start with comparing our predictions to the results from biological experiments on microbial organisms. Firstly, we will use data from Escherichia coli as it has one of the furthest developed genome-scale models and a large amount of available data in the literature [51]. Secondly, we will compare our results to predictions from other machine learning based approaches for predicting metabolic features of cancer cells, such as the recently published MetOncoFit, developed by Oruganty et al. [52]. We will begin with this part because of the availability of biological data and results from extensive FBA analysis [53].

Next, we will extend the machine learning pipeline to perform predictions of double gene deletions. More specifically, we will analyse synthetic lethality which arises when cells can survive the deletion of individual genes, but their joint deletion is lethal. This is an important problem for biomedical applications such as the discovery of new drug targets in cancer or new antibiotics against microbial infections. Furthermore, this problem is a lot more complex and FBA predictions only have an accuracy of 25 to 50% [54].

To conclude, there are a variety of areas of future study, some of which we will cover in the second part of this project.

# Bibliography

[1] Limin Li et al. Predicting enzyme targets for cancer drugs by profiling human Metabolic reactions in NCI-60 cell lines. *BMC Bioinformatics*, 11, 2010.

[2] G. Steven Martin. Cell signaling and cancer. *Cancer Cell*, 4(3):167–174, 2003.

[3] Phelan. Emerging metabolic hallmarks of cancer. *Physiology & behavior*, 176(1):139–148, 2018.

[4] Keren Yizhak et al. Phenotype-based cell-specific metabolic modeling reveals metabolic liabilities of cancer. *eLife*, 3:1–23, 2014.

[5] Gholamreza Bidkhori et al. Metabolic network-based identification and prioritization of anticancer targets based on expression data in hepatocellular carcinoma. *Frontiers in Physiology*, 9(JUL):1–11, 2018.

[6] Albert László Barabási et al. Network medicine: A network-based approach to human disease. *Nature Reviews Genetics*, 12(1):56–68, 2011.

[7] Susanna Bazzani. Promise and reality in the expanding field of network interaction analysis: Metabolic networks. *Bioinformatics and Biology Insights*, 8(phenotype I):83–91, 2014.

[8] Marco Terzer et al. Genome-scale metabolic networks. *WIREs Systems Biology and Medicine*, 1(3):285–297, 2009.

[9] Mariano Beguerisse-Díaz et al. Flux-dependent graphs for metabolic networks. *npj Systems Biology and Applications*, 4(1), 2018.

[10] R. Mahadevan and B. O. Palsson. Properties of metabolic networks: Structure versus function. *Biophysical Journal*, 88(1):7–9, 2005.

[11] Keith Henderson et al. RolX: Structural role extraction & mining in large graphs. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1231–1239, 2012.

[12] Kathryn Cooper and Mauricio Barahona. Role-based similarity in directed networks. *arXiv:1012.2726v1*, 2010.

[13] Daniel Kaslovsky. GraphRole: Automatic feature extraction and node role assignment for transfer learning on graphs. https://github.com/dkaslovsky/GraphRole.git, Accessed 15 Mar 2021.

[14] Niels Bols et al. Cellular, Molecular, Genomics, and Biomedical Approaches | Culture of Fish Cell Lines. In *Encyclopedia of Fish Physiology*, pages 1965–1970. Academic Press, 2011.

[15] Wang-Shick Ryu. Diagnosis and Methods. *Molecular Virology of Human Pathogenic Viruses*, pages 47–62, 2017.

[16] Peppino Mirabelli et al. Cancer cell lines are useful model systems for medical research. *Cancers*, 11(8), 2019.

[17] American Type Culture Collection. BT-549 (ATCC® HTB-122™). www.atcc.org/products/all/HTB-122. Accessed 26 Mar 2021.

[18] American Type Culture Collection. HCT 116 (ATCC® CCL-247™). www.atcc.org/products/all/CCL-247. Accessed 26 Mar 2021.

[19] American Type Culture Collection. K-562 (ATCC® CCL-243™). www.atcc.org/products/all/CCL-243. Accessed 26 Mar 2021.

[20] American Type Culture Collection. MCF7 (ATCC® HTB-22™). www.atcc.org/products/all/HTB-22. Accessed 26 Mar 2021.

[21] EMD Millipore Corporation. OVCAR-5 Human Cancer Cell Line (SCC259). www.merckmillipore.com/GB/en/product/OVCAR-5-Human-Cancer-Cell-Line,MM_NF-SCC259. Accessed 26 Mar 2021.

[22] Arash Iranzadeh and Nicola Jane Mulder. Bacterial Pan-Genomics. *Microbial Genomics in Sustainable Agroecosystems*, pages 21–38, 2019.

[23] Tuure Hameri et al. Kinetic models of metabolism that consider alternative steady-state solutions of intracellular fluxes and concentrations. *Metabolic Engineering*, 52:29–41, 2019.

[24] Changdai Gu et al. Current status and applications of genome-scale metabolic models. *Genome Biology*, 20(1):1–18, 2019.

[25] Zachary A. King et al. BiGG Models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic Acids Research*, 44(D1):D515–D522, 10 2015.

[26] Mihai Glont et al. BioModels: expanding horizons to include more modelling approaches and formats. *Nucleic Acids Research*, 46(D1):D1248–D1253, 11 2017.

[27] Jeffrey D. Orth et al. What is flux balance analysis? *Nature Biotechnology*, 28(3):245–248, 2010.

[28] Jan Schellenberger et al. Quantitative prediction of cellular metabolism with constraint-based models: The COBRA Toolbox v2.0. *Nature Protocols*, 6(9):1290–1307, 2011.

[29] Scott A. Becker et al. Quantitative prediction of cellular metabolism with constraint-based models: The COBRA Toolbox. *Nature Protocols*, 2(3):727–738, 2007.

[30] Ali Ebrahim et al. COBRApy: COnstraints-Based Reconstruction and Analysis for Python. *BMC Systems Biology*, 7, 2013.

[31] Andrew Makhorin. GLPK - GNU Linear Programming Kit. *Department for Applied Informatics, Moscow Aviation Institute, Moscow, Russia*, 2001.

[32] LLC. Gurobi Optimization. Gurobi Optimizer. http://www.gurobi.com.

[33] O. Mason and M. Verwoerd. Graph theory and networks in biology. *IET Systems Biology*, 1(2):89–119, 2007.

[34] Sergey Brin and Lawrence Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 56(18):3825–3833, 2012.

[35] Derek L. Hansen et al. Social network analysis: Measuring, mapping, and modeling collections of connections. *Analyzing Social Media Networks with NodeXL*, pages 31–51, 2020.

[36] Venu Satuluri et al. Markov Clustering of protein interaction networks with improved balance and scalability. *2010 ACM International Conference on Bioinformatics and Computational Biology, ACM-BCB 2010*, pages 247–256, 2010.

[37] Stijn van Dongen. Graph Clustering by Flow Simulation. *Universiteit Utrecht*, 2000.

[38] R. A. Rossi and N. K. Ahmed. Role discovery in networks. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):1112–1131, 2015.

[39] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[40] Manuel Fernández-Delgado et al. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15:3133–3181, 2014.

[41] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies*, page 3–24. IOS Press, 2007.

[42] Stephan Dreiseitl and Lucila Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics*, 35(5):352–359, 2002.

[43] Jair Cervantes et al. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408:189–215, 2020.

[44] openCOBRA. Open-source, community-developed code base for constraint-based reconstruction and analysis. http://opencobra.github.io. Accessed 20 Mar 2021.

[45] Stack Overflow. Stack Overflow Survey 2020. https://insights.stackoverflow.com/survey/2020/, Accessed 5 Mar 2021.

[46] Aric A. Hagberg et al. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[47] Gephi. Gephi - The Open Graph Viz Platform. https://gephi.org.

[48] Guy Allard. Markov clustering: Implementation of the Markov clustering (MCL) algorithm in python. https://github.com/guyallard/markov_clustering.git, 2018. Accessed 6 Feb 2021.

[49] Keith Henderson et al. It's who you know: Graph mining using recursive structural features. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 663–671, 2011.

[50] Karthik Raman and Nagasuma Chandra. Flux balance analysis of biological systems: applications and challenges. *Briefings in Bioinformatics*, 10(4):435–449, 03 2009.

[51] Jonathan M. Monk et al. iML1515, a knowledgebase that computes Escherichia coli traits. *Nature Biotechnology*, 35(10):904–908, 2017.

[52] Krishnadev Oruganty et al. Common biochemical properties of metabolic genes recurrently dysregulated in tumors. *Cancer & Metabolism*, 8(1):5, 2020.

[53] Patrick F Suthers et al. Genome-scale gene/reaction essentiality and synthetic lethality analysis. *Molecular Systems Biology*, 5(1):301, 2009.

[54] Ramy K. Aziz et al. Systems biology-guided identification of synthetic lethal gene pairs and its potential use to discover antibiotic combinations. *Scientific Reports*, 5(1):16025, 2015.