Machine learning for prediction of gene essentiality in metabolic networks

Lilli Johanna Freischem

MInf Project (Part 2) Report
Master of Informatics
School of Informatics
University of Edinburgh

2022

Abstract

The identification of essential genes is a fundamental problem in systems biology. Computational methods utilising Flux Balance Analysis (FBA) are widely used to guide biological experiments in the search for new essential genes. However, this method assumes that cells are optimised for maximal growth in every genetic state; it is unclear if this is true or if cells shift their objective to focus on survival when specific genes are knocked out.

To overcome this issue, we propose a new method for the computational prediction of essential genes. We generate Mass Flow Graphs from metabolic networks and extract structural features from the graphs; these are given to machine learning algorithms for essentiality prediction. Our proposed method predicts essential genes with near state-of-the-art accuracy, as shown by our comparison with FBA predictions. The results of this work have been submitted for publication to the 9th International Conference on the Foundations of Systems Biology in Engineering to be held in Cambridge, Massachusetts, in August 2022.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Lilli Johanna Freischem*)

Acknowledgements

First of all, I would like to thank my supervisor Diego Oyarzún for his invaluable support and guidance throughout the last two years, for giving me the opportunity to explore and learn about systems biology, and for enabling me to have such a great first research experience.

I would also like to thank my family for their love and support throughout my time at university and my friends who make Edinburgh my home away from home.

Table of Contents

1 Introduction **1**

1.1 Project Description . . . 1

1.2 Motivation . . . 1

1.3 Research Question . . . 2

1.4 Achievements . . . 3

2 Background **4**

2.1 Computational analysis of cell metabolism . . . 4

2.1.1 Genome-Scale Metabolic Models . . . 4

2.1.2 Flux Balance Analysis . . . 5

2.1.3 Mass Flow Graphs . . . 5

2.1.4 Gene-Protein-Reaction Rules . . . 7

2.1.5 Gene Essentiality . . . 8

2.1.6 COBRApy . . . 8

2.2 *Escherichia coli* model *i*ML1515 . . . 8

2.3 Machine learning for essentiality prediction . . . 9

2.4 Previous work carried out . . . 9

2.4.1 MFGpy . . . 9

2.4.2 Year 1: Predicting FBA Essentiality . . . 9

2.4.3 Year 2: Predicting Measured Essentiality . . . 10

3 Data **11**

3.1 Mass flow graph of *i*ML1515 . . . 11

3.2 Feature Extraction . . . 12

3.2.1 Adjacency Features . . . 13

3.2.2 Recursive Feature Extractor . . . 13

3.2.3 Flow Profiles . . . 14

3.3 Essentiality labels . . . 14

3.3.1 Algorithm for gene to reaction essentiality mapping . . . 15

3.4 Essentiality Analysis . . . 16

4 Methodology **17**

4.1 Overview . . . 17

4.2 Binary classification . . . 18

4.3 Preprocessing . . . 19

4.3.1 Standardisation . . . 19

4.3.2 Dimensionality Reduction 19
4.4 Evaluation metrics 21
4.4.1 Normalised confusion matrix 21
4.4.2 Precision-Recall Curve 22

5 Experiments 23
5.1 Baseline 23
5.1.1 Adjacency Features $\mathbf{X}_{\mathbf{M}}$ 24
5.1.2 ReFeX Features $\mathbf{X}_{\mathrm{ReFeX}}$ 24
5.1.3 Flow Profiles $\mathbf{X}_{FP}$ 25
5.2 Preprocessing 25
5.2.1 Adjacency Features 26
5.2.2 ReFeX 28
5.3 Hyperparameter Tuning 28
5.3.1 Adjacency Matrix 29
5.3.2 ReFeX 29
5.4 Transfer Learning 31

6 Results 32
6.1 Essentiality Analysis 32
6.2 Essentiality predictions 33
6.2.1 Standardised Adjacency Matrix 33
6.2.2 ReFeX 34
6.2.3 Comparison to essentiality predictions of *i*ML1515 35
6.3 Transfer Learning 36

7 Conclusions 37
7.1 Findings 37
7.2 Answering the Research Question 38
7.3 Evaluation 38
7.4 Future Work 39

Bibliography 40

A Appendix 44
A.1 Mapping gene essentiality to reaction essentiality 44

Acronyms

E. coli *Escherichia coli*. iv, 2–4, 7, 8, 10–12, 14–16, 33, 35, 37, 38

***i*ML1515** *Escherichia coli i*ML1515 metabolic model. iv, v, 2, 3, 8, 11, 12, 14–16, 19, 31–33, 35–39, 44

COBRA Constraint-Based Reconstruction and Analysis. 3, 8

CV Cross-Validation. 23–26, 28, 29, 33

FBA Flux Balance Analysis. i, iv, 5, 6, 9, 37

GPR Gene-Protein-Reaction. iv, 2, 7, 8, 14, 15, 44

GSMM Genome-Scale Metabolic Model. 4, 5, 7, 8

MFG Mass Flow Graph. 3, 6, 9, 18, 31, 37, 38

ML machine learning. 11

PCA Principal Component Analysis. 19, 20, 26

PR curve Precision-Recall curve. 22, 25–27, 36

ReFeX Recursive Feature eXtraction. 12–14, 23–25, 31

RF Random Forest. 9

SPCA Sparse Principal Component Analysis. 19, 20, 26

SVC c-Support Vector Machine. 18, 19

Chapter 1

Introduction

1.1 Project Description

The identification of essential genes is a fundamental problem in systems biology. Because of the high cost associated with experimental studies of gene essentiality, computational methods are widely used. The most prominent approach is Flux Balance Analysis (FBA) which has been able to accurately predict essentiality in a variety of cells. However, this method has important limitations as it assumes that organisms are optimised for maximal growth in every genetic state. Recent work has recognised the potential of combining machine learning methods with FBA to overcome this issue (Aromolaran et al., 2021).

In this project, we explore whether simple machine learning algorithms can predict metabolic essentiality using structural features extracted from metabolic graphs. More specifically, we utilise the novel Mass Flow Graph algorithm to generate graphs from metabolic networks and approach essentiality prediction as binary classification task on graph node features. In the first part of this project, we developed a machine learning pipeline to predict essentiality values obtained from FBA simulations. Our method achieved promising results with a binary prediction accuracy of 89.9%. However, the essentiality labels stem from FBA simulations, so even a perfect classifier could predict essentiality only as well as FBA. In this project, we adapt our method to use measured data from gene essentiality studies and aim to thereby overcome the issues related to using FBA values.

1.2 Motivation

A gene is considered essential if it is required for the survival of a cell or an organism (Rancati et al., 2018). Essential genes play a fundamental role in metabolic engineering, where key metabolic genes are added or deleted to redirect metabolic fluxes towards desired end products, and for therapeutic applications, where essential genes are known to represent current and potential novel drug targets (Rancati et al., 2018).

Identifying such essential genes typically requires genome-wide screening using high-throughput techniques such as RNAi or CRISPR-Cas9 (Rancati et al., 2018). However, such screens are labour-intensive and are associated with very high costs (Lu et al., 2014). Consequently, computational methods have been developed to predict essential genes and minimize the resources required for essentiality assays (Aromolaran et al., 2021; Lu et al., 2014).

Cell metabolism can be computationally represented by metabolic networks which comprise of a set of metabolites intertwined by biochemical reactions. The most popular method for the computational study of gene essentiality is Flux Balance Analysis (FBA) which computes genome-wide metabolic flux distributions from a genome-scale metabolic network (Orth et al., 2010). FBA enables simulating the cell under different genetic and environmental conditions based on the assumption that evolution has optimised cells for a certain metabolic task, typically maximal growth. This allows large-scale simulations of gene deletions which have been found to accurately predict gene essentiality in simple organisms (Monk et al., 2017). However, the validity of this assumption of optimality is not clear. In particular, the deactivation of specific genes might shift the goal of an organism to focus on survival rather than optimal growth (Montezano et al., 2015).

To overcome this issue, we explore whether machine learning models can predict essentiality using graph features derived from the genome-scale metabolic network of the wild-type cell. The wild type of an organism is its natural form without any gene deletions. More specifically, we use binary classifiers on structural features extracted from Mass Flow Graphs (MFGs) to predict gene essentiality. MFGs are reaction-based metabolic graphs that take the biological and environmental context of the cell into account (Beguerisse-Díaz et al., 2018). They are computed from the metabolic network and the wild-type FBA solution and therefore only assume that cells aim to maximise growth in their native state.

First, we present an algorithm that maps gene essentiality data to reaction essentiality labels, as MFGs are reaction-centric and nodes features are therefore reaction features. Then, we create a small dataset of reaction nodes and their essentiality labels, utilising data from experimental gene essentiality studies conducted on *Escherichia coli* (Monk et al., 2017); this dataset was used to train and validate our classification algorithms.

1.3 Research Question

This project investigates the following research question:

> Can classification algorithms predict the biological essentiality of metabolic genes using graphs derived from wild-type flux vectors?

For this purpose, we develop an algorithm to map metabolic essentiality measurements from the gene-space to the reaction-space based on Boolean Gene-Protein-Reaction relationships included within genome-scale metabolic networks. With this algorithm, we compute the essentiality of the metabolic reactions in the most complete metabolic reconstruction of *Escherichia coli*, *i*ML1515 (Monk et al., 2017). We then compute the MFG of *i*ML1515 and explore whether binary classifiers can predict reaction essentiality using node features extracted from this graph.

1.4 Achievements

Here, the main achievements of this project are summarised.

1. 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-Díaz et al. (2018).
2. Simulated cells according to biological scenarios from wet lab experiments. Reproduced published essentiality data to verify the model.
3. Developed an algorithm to translate measurements of gene essentiality to reaction essentiality.
4. Created a small dataset of reactions in the MFG of the bacterium *Escherichia coli*, computed from its metabolic reconstruction *i*ML1515, and the corresponding essentiality labels.
5. Developed a machine learning (ML) pipeline for metabolic essentiality prediction.
6. Conducted a series of experiments applying the ML pipeline to our dataset.
    (a) Compared a selection of feature sets, different methods for preprocessing of features, and an array of machine learning models for binary essentiality prediction.

    (b) Performed hyperparameter tuning on the most promising models after the initial experiments.

    (c) Evaluated the final models on a held-out test set of reactions. The best classifiers achieved a predictive performance close to FBA.
7. Compiled the results of our experiments into a paper which was submitted for publication in the Proceedings of the 9th International Conference on the Foundations of Systems Biology in Engineering (FOSBE 2022). A preprint is available at `https://www.biorxiv.org/content/10.1101/2022.03.31.486520v1`.

Chapter 2

Background

This chapter provides a literature review of the methods applied to analyse metabolic networks using machine learning. First, we introduce computational models of cell metabolism and define gene essentiality. Then, the model organism *Escherichia coli* used in our experiments is briefly introduced. Finally, we provide an overview of previous work using machine learning for essentiality prediction, both in the literature part one of this project.

2.1 Computational analysis of cell metabolism

The analysis of cell metabolism in biological experiments involves expensive and time-consuming technologies (Dong et al., 2018). This has motivated the development of a wide array of computational methods. Here, we introduce the core concepts of constraint-based modelling of metabolic networks.

2.1.1 Genome-Scale Metabolic Models

Genome-scale metabolic models (GSMMs) computationally describe the metabolism of an organism. A metabolic network that contains n metabolites and m reactions is defined 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 α_{ij} and β_{ij} are the stoichiometric coefficients of metabolite X_i in reaction R_j. This means, R_j consumes α_{ij} and produces β_{ij} molecules of metabolite X_i when it takes place. The stoichiometric coefficients can be compiled into the $n \times m$ stoichiometric matrix $\mathbf{S}$ such that 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 (Figure 2.1a, b).

Fundamental for GSMMs is the identification and mathematical definition of constraints as upper and lower flux bounds of reactions (Terzer et al., 2009). They constrain the availability of nutrients and other metabolites to define realistic metabolic behaviour (Hameri et al., 2019).

2.1.2 Flux Balance Analysis

Flux Balance Analysis (FBA) is the most widespread computational method for analysing cell metabolism (Beguerisse-Díaz et al., 2018). It is a linear programming algorithm that computes a cell's optimal flux distribution at steady state; the flux distribution defines the cell phenotype (Orth et al., 2010).

The input is a genome-scale metabolic network (Figure 2.1a). A linear system of equations is derived from its stoichiometric matrix **S** and upper and lower bounds on reaction fluxes (Figure 2.1b, c). Additionally, the cell objective function Z is defined (Figure 2.1d) which encodes the cell's biological goal. This is based on the assumption that cell metabolism is optimised to achieve maximal growth. Using linear programming, FBA finds the solution vector $\mathbf{v}^*$ to the following constrained optimisation problem:

$$\begin{aligned} \text{maximise:} \quad & Z = \mathbf{c}^\top \mathbf{v} \\ \text{subject to} \quad & \begin{cases} \frac{d\mathbf{x}}{dt} = \mathbf{S}\mathbf{v} = \mathbf{0} \\ \mathbf{v}_{lb} \le \mathbf{v} \le \mathbf{v}_{ub}, \end{cases} \end{aligned} \tag{2.2}$$

where **c** encodes the cell objective function and $\mathbf{v}_{lb}$ and $\mathbf{v}_{ub}$ are vectors containing the lower and upper bounds on reaction fluxes, respectively (Figure 2.1e). Researchers can use FBA to simulate cells under different environmental and genetic conditions by altering the reaction flux bounds (Orth et al., 2010).

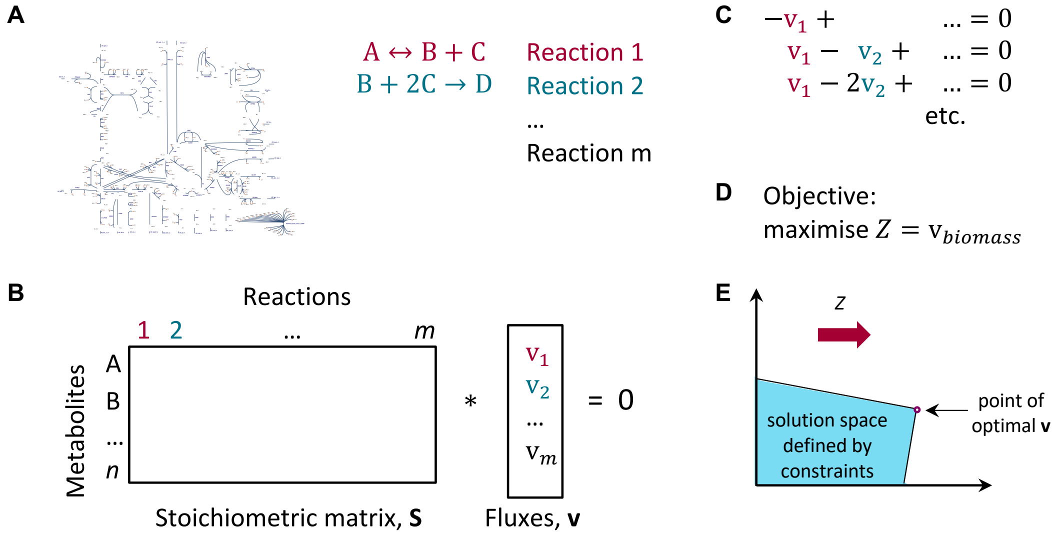


Figure 2.1: **Formulation of an FBA problem.** (**A**) A genome-scale metabolic network is reconstructed. (**B**) Metabolic reactions and constraints are mathematically represented. (**C**) A set of linear equations is defined by the mass balance (**Sv** = **0**). (**D**) An objective function Z is defined. (**E**) Fluxes that maximise Z are calculated. Figure adapted from (Orth et al., 2010).

2.1.3 Mass Flow Graphs

Researchers have analysed the structural properties of metabolic networks by applying tools from graph theory to graphs derived from GSMMs (Beguerisse-Díaz et al., 2018). Beguerisse-Diaz et al. developed an algorithm for deriving flux-based, weighted digraphs from metabolic networks called Mass Flow Graphs (MFGs). The nodes in an MFG are reactions which have edges between them if they share metabolites.

A cell's MFG is derived from its stoichiometric matrix $\mathbf{S}$ and an FBA solution vector $\mathbf{v}^*$ (Figure 2.2). By incorporating the FBA solution into the graph structure, MFGs account for environmental conditions and genetic perturbations (Beguerisse-Díaz et al., 2018).

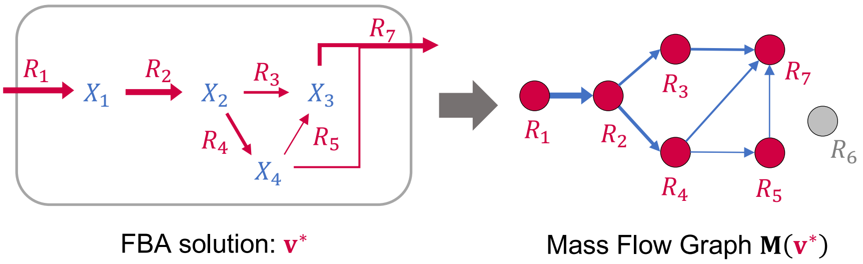


Figure 2.2: Derivation of Mass Flow Graphs from an FBA solution of a metabolic network.

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

$$\mathbf{v}^*_{2m} = \begin{bmatrix} \mathbf{v}^{*+} \\ \mathbf{v}^{*-} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \text{abs}(\mathbf{v}^*) + \mathbf{v}^* \\ \text{abs}(\mathbf{v}^*) - \mathbf{v}^* \end{bmatrix}. \tag{2.3}$$

Similarly, $\mathbf{S}$ is unfolded as:

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

where $\mathbf{r}$ is the m-dimensional reversibility vector such that $r_j = 1$ if reaction R_j is reversible and $r_j = 0$, otherwise. The $m \times m$ matrix $\text{diag}(\mathbf{r})$ contains $\mathbf{r}$ in its main diagonal.

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

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

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 equal due to the steady state condition.

Finally, the adjacency matrix of the MFG is computed as:

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

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

2.1.4 Gene-Protein-Reaction Rules

In addition to the stoichiometric matrix, GSMMs define the relationship between genes, proteins and reactions via logical rules, the Gene-Protein-Reaction (GPR) rules (Cardoso et al., 2012). As shown in Figure 2.3, GPRs provide an explicit connection between genotype and phenotype, linking the gene to the protein that catalyses a reaction in the network (Monk et al., 2017).

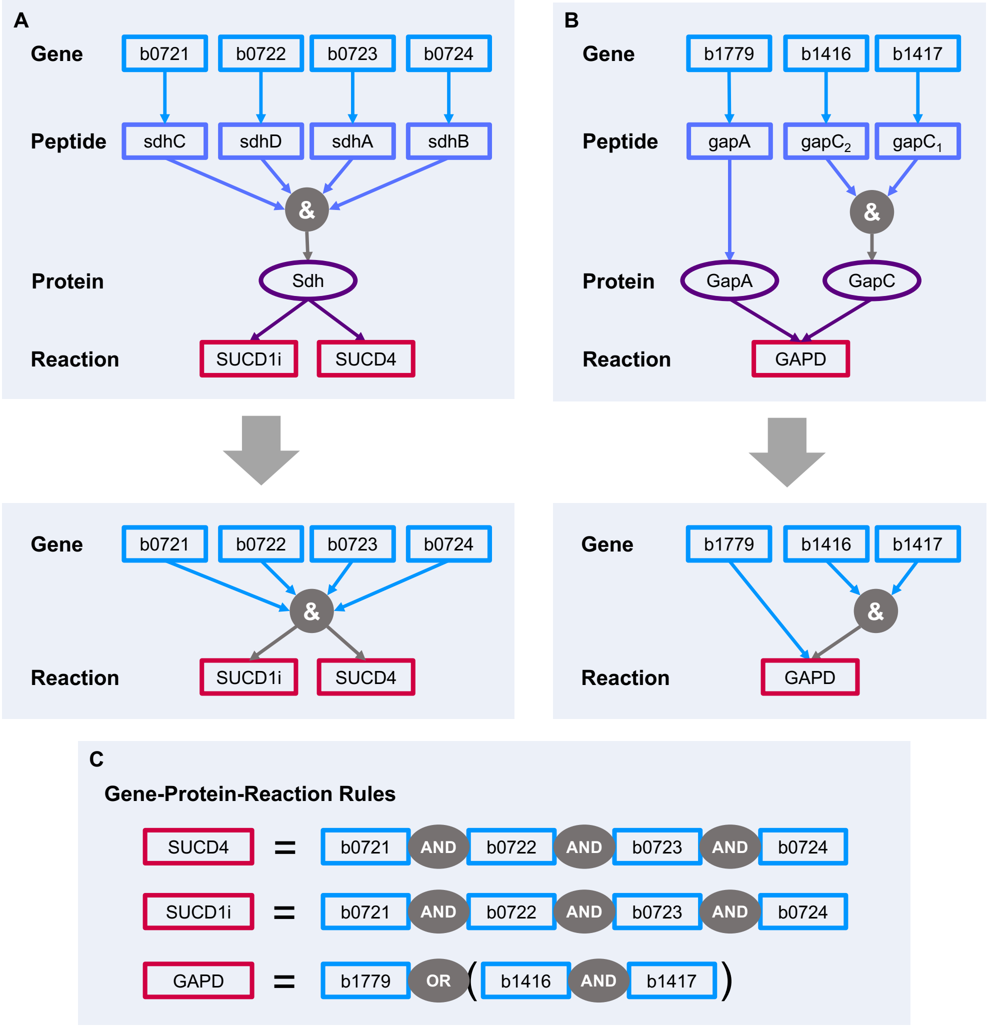


Figure 2.3: **Illustration of two Gene-Protein-Reaction (GPR) rules in *Escherichia coli*. (A)** The Sdh enzyme is built from 4 peptides and catalyses the two reactions SUCD4 and SUCD1i. **(B)** GAPD reaction is catalysed by two proteins (GapA and GapC); GapC is composed of two peptides encoded by distinct genes. **(C)** The resulting GPR rules. Figure adapted from Cardoso et al. (2012).

Including GPR rules within GSMMs is essential to enable predicting the cell phenotype under different genetic conditions, e.g., due to gene knockouts; these predictions are fundamental for gene essentiality studies (Cardoso et al., 2012).

2.1.5 Gene Essentiality

The essentiality of a gene describes how its knockout, the deactivation of that gene, affects the growth of a biological cell or organism. It is typically computed from growth rate measurements obtained from *in vitro* experiments. Given the growth rate of the wild type - the cell in its natural form without any gene deletions - g_{WT}, and the growth rate after the knockout of gene i, g_i, the essentiality of gene i is defined as:

$$e_i = 1 - \frac{g_i}{g_{WT}}. \tag{2.7}$$

By this definition, a gene has an essentiality of 1 if its knockout leads to the death of the cell, and an essentiality of 0 if it does not affect the cell's growth rate.

Most genes have an essentiality close to 0 or 1, so essentiality values are typically binarised (Bartha et al., 2018) yielding essentiality class labels:

$$y_i = \begin{cases} 0, & \text{if } e_i < 0.5 \\ 1, & \text{otherwise.} \end{cases} \tag{2.8}$$

2.1.6 COBRApy

Becker et al. developed the COBRA toolbox, a leading software package providing methods for constraint-based generation and analysis of GSMMs in MATLAB (Becker et al., 2007). Based on the COBRA toolbox, Ebrahim et al. presented COBRA for Python (COBRApy), an object-oriented framework providing methods for constraint-based generation and analysis of GSMMs (Ebrahim et al., 2013). It was developed 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.

2.2 *Escherichia coli* model *i*ML1515

The microorganism *Escherichia coli* (*E. coli*) can be found in the intestine of humans and other mammals (Kaper et al., 2004). It is the best characterised bacterial species; it was one of the first bacteria to have its complete genome decoded and has been the focus of numerous genomic studies (Sahl et al., 2013; Monk et al., 2017).

The findings of such studies have been used to develop *i*ML1515, the most complete genome-scale reconstruction of *E. coli*'s metabolic network to date (Monk et al., 2017). It includes the most up-to-date set of characterised genes and metabolic reactions for *E. coli*, as well as the corresponding Gene-Protein-Reaction relationships. *i*ML1515 has been used to build models of *E. coli* and predict their metabolic capabilities; it has been extensively validated and customized for the use in different growth conditions (Monk et al., 2017).

2.3 Machine learning for essentiality prediction

The potential of machine learning to complement constraint-based computational and experimental methods to minimise resources required for essentiality assays has been established (Zampieri et al., 2019). Early studies combining machine learning with FBA predictions utilised an array of features on top of predicted fluxes to improve essentiality predictions (Plaimas et al., 2010). Chen and Xu (2005) first applied machine learning methods to essentiality prediction; they used neural networks and support vector machines on high-throughput data to predict protein dispensability in yeast. Since then, several studies have used machine learning on computational models of cell metabolism and have shown promising predictive capabilities (Yuan et al., 2012; Guo et al., 2017; Dong et al., 2018).

Nonetheless, current studies still have important limitations. Aromolaran et al. (2021) identified the lack of data relevant for essentiality prediction as a major limiting factor. Further key issues are finding features that are relevant for essentiality prediction, improving generalisability of models across organisms, and retrieving biologically accurate essentiality measurements to use as training data (Aromolaran et al., 2021).

2.4 Previous work carried out

The first part of this project focused on predicting the FBA essentiality of reactions in cancer cells. A key contribution was the development of MFGpy, a Python package automating the generation of MFGs from COBRA models. Here, MFGpy is introduced alongside core findings that are relevant for this year's project.

2.4.1 MFGpy

In MFGpy, we implemented the novel MFG algorithm using COBRApy methods for constraint-based modelling. It contains the MFG class which provides methods to analyse, visualise, cluster and export the MFG. Importantly, after generating the MFG this package enables exporting its adjacency matrix as NumPy file or the nodes and edges tables as CSV files to be used for experiments and further analysis.

2.4.2 Year 1: Predicting FBA Essentiality

The focus of last year's work was essentiality classification of reactions in cancer cell lines based on structural features of MFGs. The MFGs were computed using MFGpy and the essentiality classes originated from FBA simulations. Multiple classification algorithms were compared; the best performance was achieved by a Random Forest which had a binary prediction accuracy of 89.9%. Our results showed that classification algorithms can accurately predict FBA reaction essentiality.

2.4.3 Year 2: Predicting Measured Essentiality

This year, we extended our analysis to the prediction of measured essentiality values by training classification algorithms on reaction essentiality data gathered from biological experiments. This was approached analogous to FBA essentiality prediction using gene essentiality data of *E. coli* (Monk et al., 2017).

Chapter 3

Data

A dataset of reactions in an MFG and their essentiality labels was required to train a supervised machine learning model to predict reaction essentiality. This dataset had to be manually constructed.

We utilised the *i*ML1515 *E. coli* model to create our dataset. Monk et al. (2017) provide the gene essentiality values that were used to design and validate *i*ML1515 together with the model. We computed the MFG of *E. coli* in the conditions matching the essentiality studies and extracted reaction-node features. Then, we translated gene essentiality measurements to reaction essentiality. The details of collection, labelling and dataset analysis are discussed in this chapter.

3.1 Mass flow graph of *i*ML1515

The objective of this project was to use biological essentiality measurements for training classification algorithms instead of the previously used FBA essentiality predictions. We therefore had to focus our analyses on an organism that has been the focus of extensive gene essentiality studies; *E. coli* satisfies this requirement.

Monk et al. provide *i*ML1515, the metabolic reconstruction of *E. coli* K-12 MG1655. Alongside the model, they provide *in vitro* gene essentiality measurements obtained from studying *E. coli* cells of the K-12 BW25113 strain. For our experiments, we used cell measurements with glucose as primary carbon source.

To train ML models to predict reaction essentiality labels inferred from the gene essentiality measurements, the computational *E. coli* model must match the cells that were investigated experimentally. We therefore had to adjust the constraints on *i*ML1515 accordingly. K-12 BW25113 lacks several genes that are present in K-12 MG1655: araBAD, rhaBAD, and lacZ (Monk et al., 2017), so we set the upper and lower flux bounds of the associated reactions to zero. The cells were growing aerobically, so we adjusted the oxygen exchange reaction bounds to simulate aerobic growth. Glucose was used as primary carbon source and, hence, the glucose uptake reaction bounds were adjusted appropriately. The updated bounds are summarised in Table 3.1.

Reactions	Adjusted bounds
L-arabinose isomerase (ARAI), L-ribulokinase (RBK_L1), Rhamnulose-1-phosphate aldolase (RMPA), Lyxose isomerase (LYXI), L-rhamnose isomerase (RMI), Rhamnulokinase (RMK), and B-galactosidase (LACZ)	$lb = ub = 0$
Oxygen exchange (EX_o2_e)	$lb = -20$
Glucose uptake (EX_glc__D_e)	$lb = -10$

Table 3.1: The adjusted lower bounds (lb) and upper bounds (ub) on reaction flux in *i*ML1515 to simulate aerobic cell growth using glucose as primary carbon source.

We calculated the FBA solution vector of the resulting model. Then, we used MFGpy to compute the corresponding MFG which contains 444 reactions (Figure 3.1).

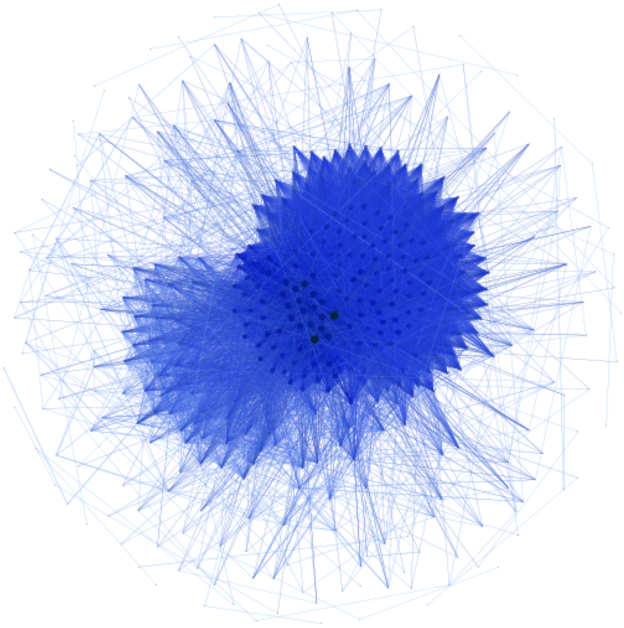


Figure 3.1: Mass flow graph of the *i*ML1515 metabolic model of *E. coli* with glucose as primary carbon source.

3.2 Feature Extraction

We extracted three distinct feature sets from the MFG of *i*ML1515, adjacency features, ReFeX features, and Flow Profiles, and investigated their suitability for essentiality prediction.

3.2.1 Adjacency Features

We obtain the first set of reaction features from the MFG adjacency matrix $\mathbf{M}$, as defined in Equation 2.6. Reactions that are not in the MFG do not contribute to the optimal FBA solution and thus are non-essential.[1] Their corresponding rows and columns in $\mathbf{M}$ only contain zeros. Hence, they are removed from $\mathbf{M}$ for classification. $\mathbf{M}_k$ denotes the resulting $k \times k$ matrix where k is the number of nodes in the MFG.

The rows in $\mathbf{M}_k$ correspond to the outgoing edges and the columns to the incoming edges of each node. To consider both incoming and outgoing fluxes for each reaction, we concatenate $\mathbf{M}_k$ and its transpose. We obtain the $k \times 2k$ node-feature matrix:

$$\mathbf{X}_\mathbf{M} = [\mathbf{M}_k \ \mathbf{M}_k^\top], \tag{3.1}$$

where row j of $\mathbf{X}$ contains first the outgoing and then the incoming edges to reaction R_j.

3.2.2 Recursive Feature Extractor

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 (Figure 3.2). Local features are measures of node degree. For a weighted, digraph, they include weighted in-, out- and total degree; egonet features consist of the weighted, directed number of edges within, entering and leaving the egonet.

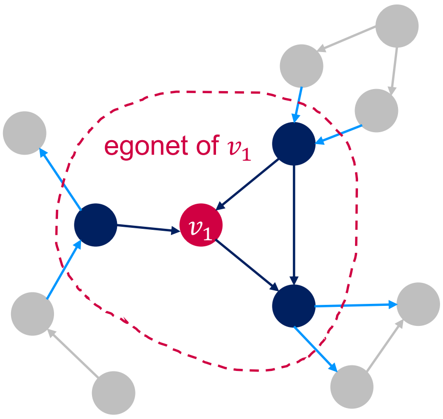


Figure 3.2: Visualisation of a node's egonet. The egonet of v_1 consists of the node itself (blue), the node's neighbours and all edges within this group (orange). In addition, ReFeX considers incoming and outgoing edges to the egonet (green).

The 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.

[1] Knocking out reactions forces their flux to be zero. Reactions that are not in the MFG already have zero flux in the wild-type cell, so their knockout does not impact cell growth meaning they are, by definition, non-essential.

There is an infinite number of possible recursive features, so 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.

We compute ReFeX features utilising Kaslovsky's implementation of the recursive feature extraction algorithm and assemble them in the $k \times f$ feature matrix:

$$\mathbf{X}_{ReFeX} = \begin{bmatrix} \mathbf{x}_{1,ReFeX} \\ \vdots \\ \mathbf{x}_{N,ReFeX} \end{bmatrix} \tag{3.2}$$

3.2.3 Flow Profiles

An alternative approach to node feature extraction from digraphs was presented by Cooper and Barahona (2010) who compute flow profiles of nodes. A node's flow profile is defined by the overall pattern of incoming and outgoing fluxes. The matrix of flow profiles $\mathbf{X}_{FP}$ is computed from the MFG adjacency matrix $\mathbf{M}$ as:

$$\mathbf{X}_{FP} = \begin{bmatrix} \mathbf{x}_{1,FP} \\ \vdots \\ \mathbf{x}_{N,FP} \end{bmatrix} \equiv \left[\cdots (\beta \mathbf{M}^T)^k \mathbf{1} \cdots \middle| \cdots (\beta \mathbf{M})^k \mathbf{1} \cdots \right] \quad \text{for } k = 1, 2, ..., \tag{3.3}$$

with $\beta = \alpha/\lambda_1$, where λ_1 is the largest eigenvalue of A and α is a scale factor between 0 and 1 that controls the weighting of the local against the global flow structure of the graph. For $\alpha \to 0$ only paths of length 1 are taken into account, so the flow profiles measure a node's in- and out-degree. For $\alpha \to 1$, the weight assigned to longer paths increases and the flow profiles increasingly take global flows into account.

3.3 Essentiality labels

Monk et al. (2017) provide gene essentiality measurements for *E. coli* K-12 BW25113 in different environmental conditions. As MFGs have reactions as nodes, we had to translate gene essentiality to reaction essentiality using the Gene-Protein-Reaction (GPR) rules included in the *i*ML1515 *E. coli* model. The GPR rules provide a formal connection that links genes to their corresponding proteins and the reactions they catalyse (Monk et al., 2017). Hence, they enable mapping data from the gene-space to the reaction-space.

*i*ML1515 provides the most up-to-date set of characterised genes and metabolic reactions for *E. coli*. It contains 1516 genes and 2712 reactions; essentiality measurements are available for 1502 of the genes (Monk et al., 2017).

3.3.1 Algorithm for gene to reaction essentiality mapping

We analysed the GPR rules of *i*ML1515 to find reaction knockouts that have a one-to-one mapping to a gene knockout under the following assumption:

Assumption 1 *If the knockout of gene G deactivates exactly one reaction R_j, the essentiality of R_j corresponds to the measured essentiality of G: $e_G = e_{R_j}$.*

This assumption formalises the following intuition: a gene knockout that deactivates a single reaction whilst all other reactions in the cell can still take place is equivalent to the knockout of that single reaction. Therefore, the cell growth rate after the knockout of G is equivalent to the growth rate after the knockout of R_j, we have $g_G = g_{R_j}$. As the essentiality of a gene or reaction U is defined as $e_U = 1 - \frac{g_U}{g_{WT}}$ we have:

$$e_G = 1 - \frac{g_G}{g_{WT}} = 1 - \frac{g_{R_j}}{g_{WT}} = e_{R_j} \tag{3.4}$$

The set of reactions with a one-to-one mapping to a gene knockout were computed from the GPR rules. Out of the 444 reactions in the MFG of *i*ML1515, only 155 reactions had a one-to-one knockout mapping to a gene (Figure 3.3). We therefore only obtain essentiality labels for 155 reactions when using one-to-one mappings only. To maximise the number of labelled reactions, we made the following second assumption:

Assumption 2 *The essentiality of reaction R_j corresponds to the measured essentiality of gene G if the knockout of G deactivates R_j. This holds even if there exists a reaction $R_l \neq R_j$ which is also deactivated by the knockout of G.*

We could increase the number essentiality labels obtained from the measurement data for reactions in the MFG to 255 when using this assumption (Figure 3.3). Hence, the advantage from using this assumption (increasing the size of our dataset by a factor of 1.65) outweighs the potential approximation error that arises. This assumption is not entirely accurate and can lead to labelling too many reactions as *essential*: (i) a reaction that is not essential but is knocked-out by the same single-gene knockout as a different, essential reaction, or (ii) a set of reactions which are not essential individually but are essential as a group and are knocked out by the same single-gene knockout. However, since we can only up/down-regulate metabolic genes and are unable to target individual reactions, the individual essentiality of reactions that can only be deactivated as a group is unclear.

Reactions that cannot be deactivated by the knockout of a single gene were excluded from model training because their essentiality labels cannot be inferred from the available growth data.

We obtain a small dataset of 255 reactions in the MFG with measured essentiality labels. The limited size is caused by the number of reactions in the MFG (only 444)

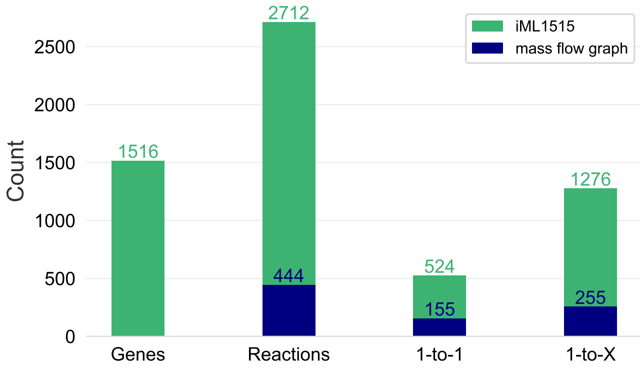


Figure 3.3: Reaction and gene counts in *i*ML1515. From left to right, we show the total number of genes, the total number of reactions, the number of reactions which are deactivated by single-gene knockouts that do not deactivate any other reactions (”1-to-1”) and that deactivate one or more reactions (”1-to-X”).

as well as the number of reactions whose essentiality cannot be measured via single gene knockouts. Nonetheless, this dataset is sufficient for our experiments which act as a proof of concept. Pseudocode for the algorithm is included in the Appendix.

3.4 Essentiality Analysis

We conducted a first analysis of the essentiality distribution in the dataset. As shown in Fig. 3.4, around 75% of labelled reactions in the MFG are essential. Only 23 reactions (9%) have an essentiality between 0.1 and 0.9. This supports the claim that essentiality prediction can be approached as binary classification problem.

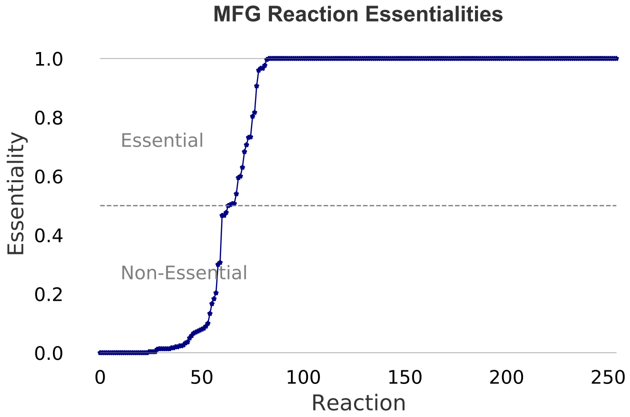


Figure 3.4: Measured essentiality of all reactions in the MFG of the K-12 BW25113 *E. coli* strain. The MFG was computed using the *i*ML1515 *E. coli* model.

Chapter 4

Methodology

In this chapter, we present the methodology used for training binary classifiers to predict reaction essentiality based on structural features extracted from MFGs. First, we provide an overview of the machine learning pipeline. Then, we introduce the four classification algorithms we used in our experiments. Next, we discuss the data preprocessing methods that were investigated. Finally, we present the evaluation metrics used to quantitatively assess and compare the performance of different classification models.

4.1 Overview

We train binary classifiers to predict reaction essentiality labels from MFG reaction node features An overview of the machine learning pipeline is depicted in Figure 4.1. Starting with an MFG adjacency matrix **M**, we extract node feature matrices (Section 3.2) and apply preprocessing steps (Section 4.3). The resulting feature matrix is provided as input to classification algorithms, together with the essentiality class labels computed from gene essentiality measurements (Section 3.3). Finally, we performed hyperparameter optimisation (Section 4.4).

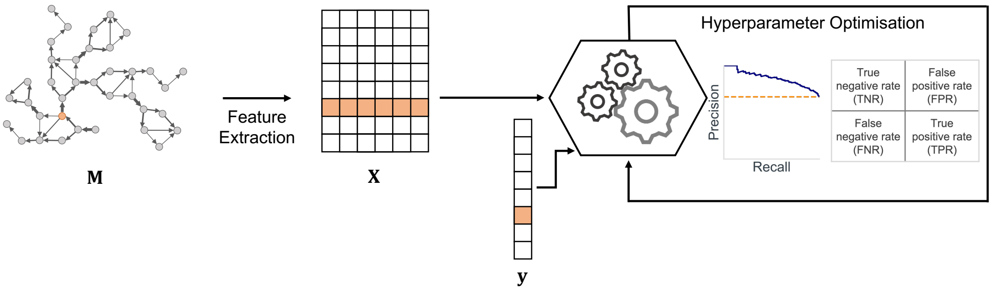


Figure 4.1: **Training binary classification algorithms on Mass Flow Graphs.** From the adjacency matrix of a Mass Flow Graph **M**, the node-feature matrix **X** is computed. Binary classifiers are trained using **X** and measured essentiality labels **y**. For hyperparameter optimisation, different performance metrics are tracked, focusing on precision and recall.

4.2 Binary classification

In a binary classification problem, a set of elements is divided into two classes. Given an element from this set, a classification algorithm (or classifier) assigns a class label according to the element's features and classification rules (Larrañaga et al., 2006).

Supervised classification algorithms automatically learn these classification rules from a set of labelled samples, the training set:

$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), ..., (\mathbf{x}^{(J)}, y^{(J)})\}, \tag{4.1}$$

which contains a d-dimensional feature vector $\mathbf{x}^{(j)} \in \mathcal{R}^d$ and a class label $y^{(j)} \in \{0, 1\}$ for each sample j.

The feature vectors and labels are assembled in a feature matrix $\mathbf{X}$ and a vector of class labels $\mathbf{y}$:

$$\begin{aligned} \mathbf{X} &= [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(J)}]^\top, \\ \mathbf{y} &= [y^{(1)}, y^{(2)}, ..., y^{(J)}]^\top, \end{aligned} \tag{4.2}$$

which are provided to the classifiers to learn the classification rules. Afterwards, it can assign labels to new instances according to their feature vectors. Given a new element's feature vector $\mathbf{x}$, it predicts the corresponding class label $\hat{y}$ based on the learned classification rules.

Various supervised classification algorithms exist; which algorithm is most suitable for a given task depends significantly on the characteristics of the dataset at hand. We experimentally explored the suitability of four standard classification algorithms, Random Forests, Logistic Regression, Multilayer Perceptron, and Support Vector Machine, for essentiality prediction from MFG features.

Random Forest (RF) is an ensemble method that trains a collection of decision trees individually, on a random subset of the training data. To classify a given example, each tree predicts its class and the RF outputs the most popular prediction.

Logistic Regression uses a logistic sigmoid function as transformation to linear output labels to model a binary output variable (Dreiseitl and Ohno-Machado, 2002). 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 (Dreiseitl and Ohno-Machado, 2002).

Multilayer Perceptrons (MLP) use feedforward neural networks to learn classification rules; they are universal function approximators and are thus able to learn non-linear models (Kotsiantis, 2007).

c-Support Vector Machines (SVC) find a hyperplane which separates the two classes in the feature space (Kotsiantis, 2007). They aim to maximise the margin that separates the classes which reduces the risk of false classification. SVCs are memory efficient and versatile classifiers which can be used for binary as well as multi-class classification.

4.3 Preprocessing

4.3.1 Standardisation

Classification models, especially distance-based classifiers such as SVCs, are negatively impacted by differently scaled dimensions in the data. They aim to maximise the distance between the separating plane and the support vectors; if features have different scales, features with large values will dominate over smaller features when calculating the distance. Prior to training such models, the dataset thus has to be standardised.

To standardise the feature matrix, the mean and standard deviation of each feature dimension is computed from the training samples. These values are used to compute standardised features as:

$$\tilde{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}, \tag{4.3}$$

where $\mathbf{x}_{ij}$ is the entry in the ith column and the jth row of $\mathbf{X}$ and μ_j and σ_j are the mean and standard deviation of column i in the training set.

If the matrix that is standardised is sparse, the mean is not subtracted to prevent destroying the sparsity structure (Zheng and Casari, 2018). Instead, features are scaled to unit standard deviation only:

$$\tilde{x}_{ij,\,(sparse)} = \frac{x_{ij}}{\sigma_j}. \tag{4.4}$$

Besides improving the performance of classifiers, standardisation is a prerequisite for using dimensionality reduction techniques such as PCA. Tree-based models, however, do not require standardisation as they find partition rules that best split the feature space which is independent of feature scaling.

4.3.2 Dimensionality Reduction

The adjacency feature matrix $\mathbf{X_M}$ is of shape $k \times 2k$ and thus high-dimensional (Section 3.2.1). Further, the matrix is sparse; only 7% of the entries in the adjacency feature matrix of the iML1515 MFG are nonzero. The number of trainable parameters in classification algorithms rises with the dimensionality of the input data. To avoid training unnecessarily complex models, we attempt to reduce the dimensionality of the input features. Each column has at least one non-zero entry, so we are not able to simply remove columns from the input matrix.

We therefore investigate the effects of adding dimensionality reduction, more specifically Principal Component Analysis and Sparse Principal Component Analysis, to the ML pipeline. The benefit would be a significant reduction in problem size which would be especially useful because of the small amount of data available for model training.

4.3.2.1 Principal Component Analysis

Principal component analysis (PCA) is a method for linear dimensionality reduction that minimises information loss and increases the interpretability of the data (Jolliffe and Cadima, 2016).

To compute the principal components of $\mathbf{X}$ we first compute its covariance matrix $\mathbf{A}$ as:

$$\mathbf{A} = cov(\mathbf{X}, \mathbf{X}) = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top. \tag{4.5}$$

The principal components are the eigenvectors of this covariance matrix, so they are the vectors $\mathbf{v}$ that solve the following equation:

$$\mathbf{A} \cdot \mathbf{v} = \lambda \cdot \mathbf{v}, \tag{4.6}$$

where λ is the corresponding eigenvalue.

The eigenvectors are sorted by their eigenvalues, as they correspond to the amount of variation that samples show along the direction of the eigenvector. To reduce the dimensionality of $\mathbf{X}$ to q, we therefore select the eigenvectors with the q largest eigenvalues.

Using PCA to reduce the dimensionality of $\mathbf{X}$, we obtain:

$$\mathbf{X}_{PCA_q} = \mathrm{PCA}_q(\mathbf{X}), \tag{4.7}$$

where q denotes the number of principal components that are kept and therefore the dimensionality of $\mathbf{X}_{PCA_q}$. It has to be chosen manually which is a trade-off between minimising dimensionality and minimising information loss (Jolliffe and Cadima, 2016).

4.3.2.2 Sparse Principal Component Analysis

Regular PCA suffers from the restriction that each principal component is a linear combination of all other variables which makes interpreting the derived components as new features difficult (Zou and Xue, 2018). Sparse variants of PCA have been developed to overcome this issue.

Zou et al. (2006) introduced Sparse Principal Component Analysis (SPCA), the first computationally efficient algorithm for computing sparse principal components. In SPCA, components are computed using the following regression formulation:

$$\begin{gathered}(U^*, V^*) = \text{ arg } \min_{U,V} \frac{1}{2} ||X - UV||^2_{\text{Fro}} + \alpha ||V||_{1,1} \\ \text{subject to } ||U_k||_2 <= 1 \text{ for all } 0 \leq k < q,\end{gathered} \tag{4.8}$$

where $||\cdot||_{\text{Fro}}$ is the Frobenius norm, $||\cdot||_{1,1}$ is the entry-wise matrix norm which is the sum of the absolute values of all the entries in the matrix, and q denotes the number of sparse components that are used (Pedregosa et al., 2011). Including the entry-wise matrix norm of V in the loss term induces sparsity in the derived components and prevents learning components from noise when a small number of training samples is available (Pedregosa et al., 2011). The hyperparameter α controls the sparsity; small values lead to a gently regularized factorization, while larger values shrink many coefficients to zero.

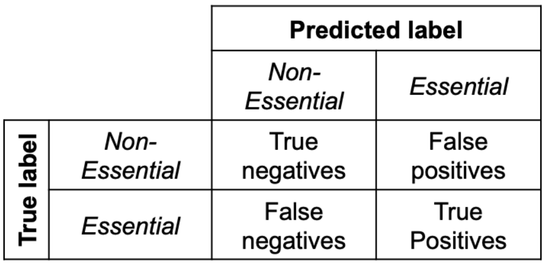
We use SPCA to reduce the dimensionality of $\mathbf{X}$ and obtain:

$$\mathbf{X}_{SPCA_q} = SPCA_q(\mathbf{X}). \tag{4.9}$$

4.4 Evaluation metrics

A binary classifier labels examples as either positive or negative. In this project, the examples are reactions that are labelled as *essential* (positive, class label 1) or *non-essential* (negative, class-label 0). The decisions made by the classifier can be represented as confusion matrix (Figure 4.2) which has the following four categories:

1. True Positives (TP): *essential* reactions correctly labelled as *essential*,
2. False Positives (FP): *non-essential* reactions incorrectly labelled as *essential*,
3. True Negatives (TN): *non-essential* reactions correctly labelled as *non-essential*,
4. False Negatives (FN): *essential* reactions incorrectly labelled as *non-essential*.



		Predicted label	
		Non-Essential	*Essential*
True label	*Non-Essential*	True negatives	False positives
	Essential	False negatives	True Positives

Figure 4.2: Confusion matrix for binary essentiality prediction.

Performance metrics need to be defined to assess classification performance. Our dataset contains an unbalanced number of essential and non-essential reactions (25% / 75% split). Overall classification accuracy therefore does not contain sufficient information to evaluate classification performance. Instead, we use performance metrics based on the confusion matrix and the precision-recall curve for evaluation.

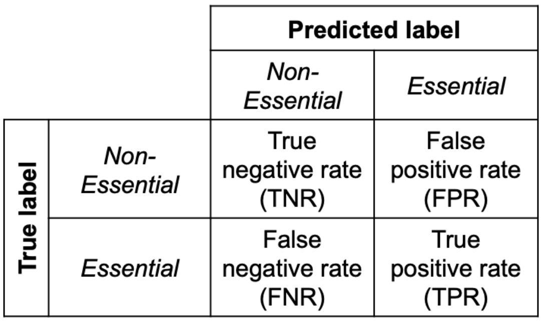
Performance was quantified using accuracy, precision, recall (true positive rate, TPR), specificity (true negative rate, TNR) and the F1 Score:

$$\begin{aligned}
\text{accuracy} &= (\text{TP} + \text{TN}) \,/\, \text{k} \\
\text{precision} &= \text{TP} \,/\, (\text{TP} + \text{FP}) \\
\text{recall (TPR)} &= \text{TP} \,/\, (\text{TP} + \text{FN}) \\
\text{specificity (TNR)} &= \text{TN} \,/\, (\text{TN} + \text{FP}) \\
\text{F1 Score} &= \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.
\end{aligned} \tag{4.10}$$

In addition, we compute the macro-averaged F1 (Macro-F1) score which is the arithmetic mean of the F1 scores of the positive and negative class. Therefore, it prevents neglecting the classification performance on non-essential reactions, the minority class.

4.4.1 Normalised confusion matrix

Because of the significant imbalance of non-essential to essential reactions, used the normalised confusion matrix to evaluated classification performance (Figure 4.3). It contains the entries of a standard confusion matrix (TP, FP, TN, FN) normalised by class size, so its rows sum to 1. This makes it easier to compare performance on the two classes.



		Predicted label	
		Non-Essential	*Essential*
True label	*Non-Essential*	True negative rate (TNR)	False positive rate (FPR)
	Essential	False negative rate (FNR)	True positive rate (TPR)

Figure 4.3: Normalised confusion matrix for binary essentiality prediction.

4.4.2 Precision-Recall Curve

The precision-recall curve (PR curve) provides a visualisation of classification performance and is well-suited for comparing classifiers on unbalanced datasets (Fu et al., 2018). In addition, it enables computing the area under the curve (AUC-PR). The AUC-PR depends on the fraction of samples that belong to the positive class. In our case, approximately 75% of reactions are essential. Hence, a no-skill classifier, which simply labels all samples as essential, will have an AUC-PR of 0.75. A perfect classifier on the other hand has an AUC-PR of 1.0. A classifier with some predictive skill will therefore have AUC-PR $\in (0.75, 1)$. The PR curve of an example classifier trained on a dataset containing 75% positive samples is shown in Fig. 4.4.

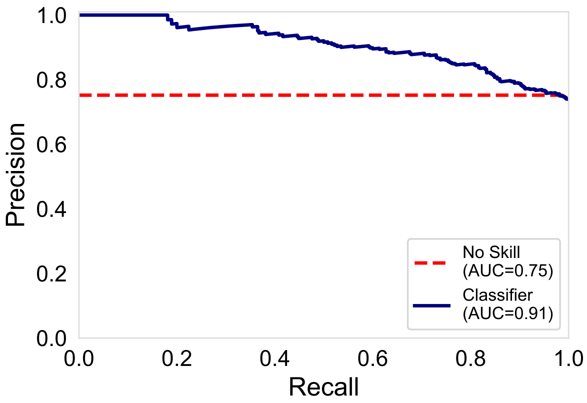


Figure 4.4: Example of a precision-recall curve with a class imbalance of 25% to 75%.

Chapter 5

Experiments

This chapter presents the essentiality prediction experiments conducted in this project alongside intermedite results which guided further steps. The aim of our experiments was to find structural features that enable simple binary classification models to predict reaction essentiality.

Investigated were three sets of features extracted from the MFG: adjacency features, Recursive Feature eXtraction (ReFeX) features, and Flow Profiles. For each feature set, we first trained simple classifiers to establish performance baselines. Following this, different preprocessing methods and their effect on model performance are investigated. Finally, hyperparameter tuning is carried out on the best models on each feature set.

Prior to our experiments, we used stratified sampling to set aside 20% of the input samples as test set. These 51 reactions were not used in training but only for the final evaluation of the best models (Section 6.2). We used stratified 5-fold cross-validation (CV) to compare different models, feature sets, and hyperparameters.

We conducted our experiments using an array of simple classification algorithms: Random Forests, Logistic Regression, Multilayer Perceptrons, and c-Support Vector Machines. The machine learning models were implemented using the Python library scikit-learn (Pedregosa et al., 2011). Unless specified otherwise, the models were trained using scikit-learn's default parameters.

5.1 Baseline

Initial experiments were conducted to gain a first insight into the suitability of the three candidate feature sets and obtain performance baselines. To enable a fair comparison of the three feature sets, model hyperparameters were kept constant across all baseline experiments. The kernel of the SVC was set to *polynomial* because classifiers that use the default *rbf* kernel were unskilled and classified all reactions as essential.

We focused on the Macro-F1 score to evaluate model performance as it best represents classification skill on our unbalanced dataset.

5.1.1 Adjacency Features $\mathbf{X_M}$

Our first baseline experiments were conducted on the adjacency features as derived in Equation 3.1. Table 5.1 shows the CV accuracy, F1 Score and Macro-F1 score of each classification algorithm. The classifier with the best Macro-F1 score of 58.3% was a Random Forest; this is our baseline model for further evaluations. Its normalised confusion matrix is displayed in Figure 5.1

To gain a deeper insight into classification performance, we investigated the confusion matrices for each classifier. We found that the Logistic Regression and the SVC model struggle to learn from the input data and classify nearly all reactions as *essential*.

Classifier	Accuracy	F1 Score	Macro-F1 Score
RF	76.0 ± 4.2%	85.3 ± 2.7%	58.3 ± 8.2%
LogReg	74.0 ± 1.9%	85.0 ± 1.2%	44.2 ± 3.5%
MLP	76.5 ± 1.9%	86.2 ± 1.0%	52.0 ± 8.3%
SVC (poly)	74.5 ± 2.3%	85.3 ± 1.5%	44.5 ± 4.1%

Table 5.1: Baseline CV performance of different classifiers on the feature matrix **X**.

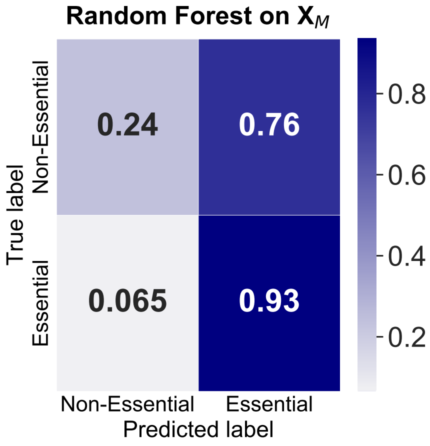


Figure 5.1: Cross-validation confusion matrix of the Random Forest on the adjacency features $\mathbf{X_M}$.

5.1.2 ReFeX Features $\mathbf{X_{ReFeX}}$

Our second set of baseline experiments was conducted on ReFeX features. Except for the SVC, all models reach higher Macro-F1 scores on average, however, the larger standard deviations indicate that performance varies stronger across CV folds. Noticeable is the significantly better performance of the logistic regression model on ReFeX features compared to adjacency features; on average, its Macro-F1 score is 14.2% higher. The performance of the Random Forest on this feature set becomes our baseline on ReFeX features with a Macro-F1 score of 61.2%.

Classifier	Accuracy	F1 Score	Macro-F1 Score
RF	76.4 ± 7.1%	85.5 ± 4.2%	61.2 ± 13.3%
LogReg	74.5 ± 1.4%	84.1 ± 1.3%	58.4 ± 5.5%
MLP	67.7 ± 7.3%	78.9 ± 5.2%	53.9 ± 10.4%
SVC (poly)	75.0 ± 0.9%	85.7 ± 0.6%	42.9 ± 0.3%

Table 5.2: Baseline CV performance of different classifiers on ReFeX features.

5.1.3 Flow Profiles X_{FP}

The third feature set we investigated were Flow Profiles (see Section 3.2.3). However, no classifier showed predictive skill when trained on this feature set. As example, Figure 5.2 shows the cross-validated PR curves of the Random Forest; the other classifiers performed similar. Flow Profiles converge by definition, so poor classification performance cannot be caused by differently scaled features even in the distance-based classifiers. We decided not to conduct further experiments on this feature set.

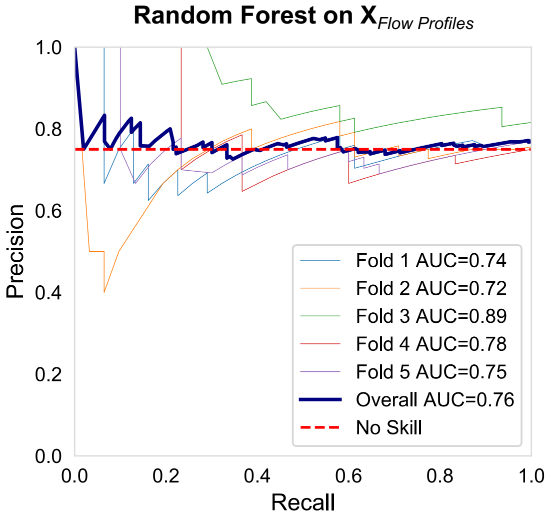


Figure 5.2: Precision-Recall curve of the Random Forest trained on Flow Profiles.

5.2 Preprocessing

After establishing performance baselines on adjacency matrix and ReFeX features, we examined whether preprocessing could improve classification performance. First, we standardised features as distance-based classification algorithms are negatively impacted by differently scaled feature dimensions. The values used for standardisation are computed from the training samples and stored to transform future inputs accordingly. Second, we applied dimensionality reduction methods to the high-dimensional adjacency features but found that classifiers cannot use the emerging features to predict reaction essentiality.

5.2.1 Adjacency Features

5.2.1.1 Standardised Adjacency Features

Standardisation is usually achieved by subtracting the feature mean and scaling to unit variance. However, as explained previously the MFG adjacency matrix tends to be sparse (Section 4.3.2). To maintain the sparsity structure in the data, we only scale features to unit variance.

Classifier	Accuracy	F1 Score	Macro-F1 Score
RF	76.0 ± 5.4%	85.2 ± 3.7%	59.4 ± 9.2%
LogReg	69.1 ± 6.8%	79.1 ± 6.1%	57.7 ± 7.6%
MLP	72.5 ± 4.1%	82.7 ± 2.6%	57.3 ± 7.5%
SVC (poly)	68.2 ± 5.9%	78.7 ± 6.3%	53.5 ± 7.6%

Table 5.3: Performance of different classifiers on the standardised feature matrix $\mathbf{X}_{\mathbf{M},std}$.

The classification performance on the standardised adjacency matrix is summarised in Table 5.3. Remarkable are the significant improvements of the Macro-F1 score of Logistic Regression, MLP, and SVC classifiers of 13.5%, 9% and 5.3%, respectively. This is not surprising since distance-based classifiers in particular struggle to learn from data with differently scaled feature dimensions. As expected, the performance of the Random Forests is not affected by scaling features and therefore remains unchanged; the minimal variations in metrics can be explained by the randomness associated with the training process. Adjacency features were standardised for all further experiments.

5.2.1.2 Dimensionality Reduction

The feature vectors in $X_{\mathbf{M}}$ are 888-dimensional because the MFG contains 444 nodes; the dimensionality of our data is over three times the size of our dataset. Therefore, we explored if dimensionality reduction techniques can reduce the problem size and improve classification performance. For this purpose, we applied Principal Component Analysis and Sparse Principal Component Analysis.

Using standard principal components as input features was found to be detrimental for performance. The resulting models could not reach CV accuracies beyond 70% and the Macro-F1 scores were around 50% and lower (Figure 5.3B).

We made the same observations when sparse principal components were used as input features; classifiers trained on sparse PCs could not reach Macro-F1 scores beyond 55% (Figure 5.4A). The highest Macro-F1 score was reached by the model trained on 22 sparse PCs; a closer investigation of its cross-validated confusion matrix and PR curve showed that this model hardly learns from the provided data (Figure 5.4B and C).

Overall, we found that classifiers cannot use the lower-dimensional features computed with (Sparse) PCA to predict reaction essentiality. Hence, we did not apply dimensionality reduction in further experiments.

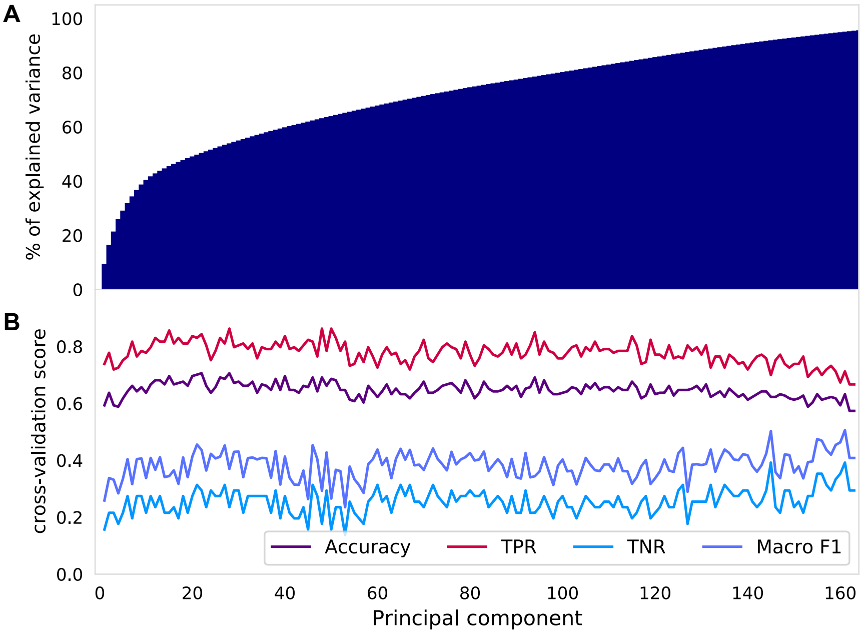


Figure 5.3: **Principal component analysis of $\mathbf{X}_M$.** (**A**) A cumulative plot of the explained variance of the principal components. (**B**) Random Forest CV classification performance across varying number of principal components.

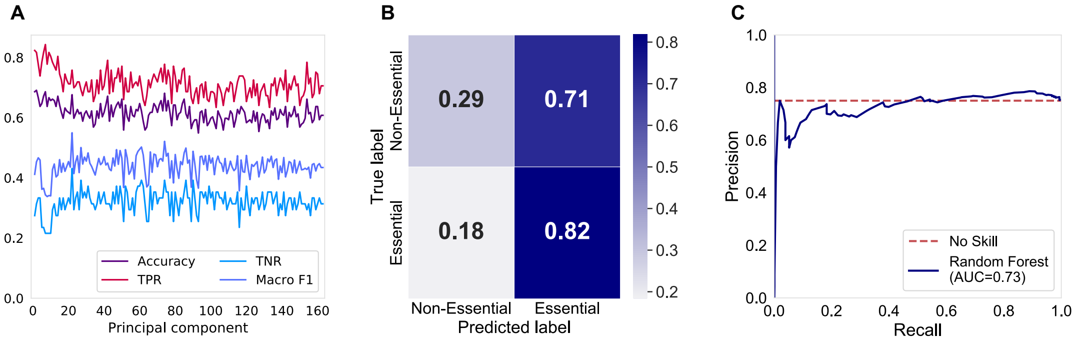


Figure 5.4: **Sparse Principal Component Analysis of $\mathbf{X}_M$.** (**A**) Random Forest CV classification performance across varying number of sparse principal components. (**B**) Cross-validated confusion matrix of Random Forest on 22 sparse PCs. (**C**) Cross-validated PR curve of Random Forest on 22 sparse PCs.

5.2.2 ReFeX

5.2.2.1 Standardised ReFeX

The ReFeX features were standardised by removing the mean from each dimension and scaling to unit variance. The mean was removed because firstly, the ReFeX feature matrix is dense and secondly, features are of very different scales and feature means range from $\sim$1 to $\sim$200,000. The vastly different scales emerge from the recursive computation of averages and sums of positive values.

Standardising ReFeX features especially improved the performance of the SVC; its Macro-F1 score improved by 9.8%. The Macro-F1 scores of both the Logistic Regression model and the MLP also improved, by 3.1% and 5.8%, respectively. As expected, the performance of the Random Forest was not affected by standardising the input features. Further experiments were conducted on standardised ReFeX features only.

Classifier	Accuracy	F1 Score	Macro-F1 Score
RF	76.0 ± 6.2%	85.2 ± 3.7%	60.4 ± 11.8%
LogReg	77.5 ± 4.9%	86.3 ± 2.8%	61.5 ± 10.0%
MLP	73.0 ± 4.8%	82.4 ± 4.1%	59.7 ± 7.0%
SVC (poly)	77.4 ± 2.9%	86.9 ± 1.6%	52.7 ± 9.4%

Table 5.4: Performance of classifiers on the standardised ReFeX features.

5.3 Hyperparameter Tuning

We investigated whether hyperparameter tuning could improve the classification performance of our best models found in the comparison of several classification algorithms and feature sets in our baseline and preprocessing experiments. Again, different hyperparameter settings were compared using stratified 5-fold CV.

To reduce the computational cost of hyperparameter optimisation, we used the hyperopt package (Bergstra et al., 2013) which employs Bayesian optimisation to automatically choose optimal hyperparameters. This method produces a reproducible and unbiased optimisation process and yields a significant reduction in computation time.

Selecting hyperparameters through Bayesian optimization requires the definition of a measure of prediction quality. As explained previously, accuracy is not a good measure of classification skill due to the class imbalance in our dataset. Empirically, we found that the hyperparameter settings found when optimising for accuracy produce unskilled classifiers which naively label all reactions as *essential*. Instead, we used the Macro-F1 score which prevents choosing models without classification skill for non-essential reactions by assigning equal weight to precision and recall on non-essential and essential reactions.

5.3.1 Adjacency Matrix

The model with the highest Macro-F1 score on standardised adjacency features was a Random Forest. We used hyperopt to find optimal hyperparameters within the search space shown in Table 5.5.

Random Forest Hyperparameters	
Hyperparameter	**Search Space**
Number of estimators	randint(100, 500)
Maximum depth of the tree	randint(10,200)
Minimum number of samples per leaf	randint(1, 10)
Criterion	gini or entropy
Max features	sqrt or log2

Table 5.5: The hyperparameter search space used for optimising the Random Forest.

The optimised model contains 300 trees, with a maximum depth of 50, using information gain as criterion, and considering $\log_2(2k)$ features when looking for the best split. In CV, it had a Macro-F1 score of 64.8%, with a precision of 82% and 86% recall. Compared to the Random Forest with default hyperparameters, the Macro-F1 score improved by 5.3%, however, the accuracy and F1 Score are slightly lower (1.1% and 1.8%). Whilst the model is better at correctly classifying non-essential reactions, it makes more mistakes in its predictions of essential reactions.

Classifier	Accuracy	F1 Score	Macro-F1 Score
RF	$74.9 \pm 8.6\%$	$83.4 \pm 6.5\%$	$64.8 \pm 10.8\%$

Table 5.6: Performance of the Random Forest with largest Macro-F1 score on standardised adjacency features found in hyperparameter tuning.

5.3.2 ReFeX

On the standardised ReFeX features, the Random Forest, MLP and Logistic Regression classifiers performed well, with Macro-F1 scores of 60-61%. We therefore decided to examine if hyperparameter tuning could improve the performance of these three classifiers.

For the Random Forest, we employed the same search space as presented in Section 5.3.1, however, model performance could not be further improved.

The search space of the Logistic Regression and the MLP classifier, as well as the chosen hyperparameters are displayed in Tables 5.7 and 5.8. Through hyperparameter search, we were able to improve the performance of both classifiers. The best model we found was the Logistic Regression classifier with a Macro-F1 score of 63.2%, an improvement of 1.7% over the model without hyperparameter tuning. The Macro-F1 score of the MLP was only slightly lower, with an average of 62.8% in cross-validation.

Logistic Regression Hyperparameters		
Hyperparameter	**Search Space**	**Chosen Value**
Penalty	l1 or l2	l1
Fit intercept	True or False	False
Class weight	None, {0:2, 1:1}, {0:3, 1:1}	{0:2, 1:1}
Solver	liblinear, lbfgs	liblinear
Tolerance	uniform($10^{-6}, 10^{-2}$)	0.0025
C	uniform(0.5, 1.5)	0.75

Table 5.7: The hyperparameter search space used for optimising the Logistic Regression classifier on standardised ReFeX Features.

Multilayer Perceptron Hyperparameters		
Hyperparameter	**Search Space**	**Chosen Value**
Learning rate	constant, invscaling, adaptive	adaptive
Hidden layer sizes	(50,100,50), (50,50,50), (25,50,25), (100), (50), (25)	(50)
Activation	identity, logistic, tanh, relu	tanh
Solver	lbfgs, adam	lbfgs
Early stopping	True, False	False
Alpha	uniform($10^{-6}, 10^{-3}$)	0.0003

Table 5.8: The hyperparameter search space used for optimising the Multilayer Perceptron classifier on standardised ReFeX features.

Classifier	Accuracy	F1 Score	Macro-F1 Score
RF	76.0 ± 6.2%	85.2 ± 3.7%	60.4 ± 11.8%
MLP	72.5 ± 6.9%	81.6 ± 4.8%	62.8 ± 9.7%
LogReg	75.5 ± 4.6%	84.3 ± 2.6%	63.2 ± 9.4%

Table 5.9: Performance of the three best classifiers with largest $F1_{macro}$ score on standardised ReFeX features found in hyperparameter tuning.

5.4 Transfer Learning

Our final experiment investigated whether classification models trained on genes with essentiality labels of one MFG could be used to predict gene essentiality in different environmental conditions. For this purpose, we generated a second MFG of *i*ML1515, now using acetate as carbon source.

Acetate was chosen as it had the highest number of genes with different measured essentiality compared to glucose. We therefore expect models to find it harder to predict gene essentialities in this environment. Hence, it is most suited for assessing the suitability of our method for predicting conditional essentiality.

For this experiment, we computed the MFGs for both environmental conditions and generated the adjacency matrix feature sets as presented in Section 3.2.1. To use models trained on the glucose MFG for predictions on the acetate MFG, the columns in the feature matrix need to correspond to the same features; in this case, to edges to the same reactions. The two MFGs contain 397 of the same reactions; 47 and 44 reactions appear only in the MFG of glucose and acetate, respectively. We removed columns that correspond to reactions which only appear in one of the MFGs from each feature matrix. Next, we computed essentiality labels for reactions in the acetate MFG as presented in Section 3.3.

We trained the Random Forest with optimised hyperparameters on the training set of the glucose reactions and used the glucose test set for validation; this ensured that removing reactions that are unique two the glucose MFG did not significantly impact classification performance. We found that classification performance did not worsen when these columns were removed from the feature matrix.

Chapter 6

Results

In this chapter, the final results of our experiments are presented. First, we present our findings from essentiality analysis in the MFG of *i*ML1515. Next, we evaluate the best essentiality classifiers on the held-out test set to gain insight into the generalisation performance of our algorithm. Finally, we compare our results to the FBA essentiality predictions of *i*ML1515 which are used as benchmark.

6.1 Essentiality Analysis

We analysed the essentiality distribution in the complete *i*ML1515 model and in the corresponding MFG.

Gene essentiality measurements are available for 1503 out of the 1516 genes included in *i*ML1515 (Monk et al., 2017). When grown on glucose as primary carbon source, 1252 genes are non-essential and 251 genes are essential, which corresponds to 16.7% of genes being essential (Figure 6.1).

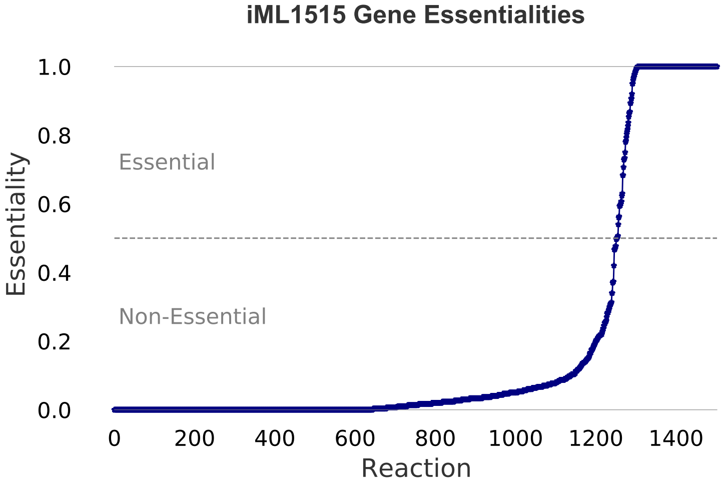


Figure 6.1: Measured gene essentiality in *i*ML1515 using glucose as carbon source, data from (Monk et al., 2017).

The essentiality distribution changes significantly when only genes which correspond to reactions included in the MFG are considered (Figure 3.4). In the MFG, we have 63 non-essential and 192 essential reactions (75.3% reactions are essential). Figure 6.2 shows the corresponding MFG with reaction nodes coloured according to their essentiality label.

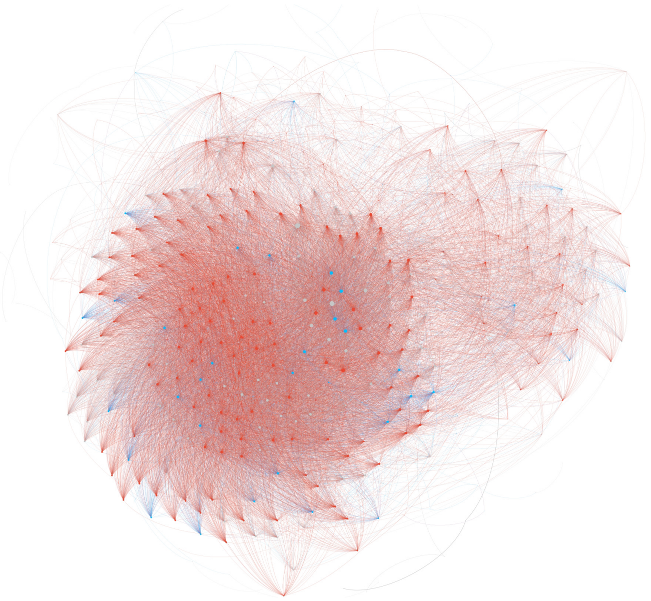


Figure 6.2: Mass flow graph of *Escherichia coli* (*E. coli*) under aerobic growth with glucose as sole carbon source, computed from *i*ML1515. The $k = 444$ nodes are coloured by essentiality label (blue=non-essential, red=essential, grey=essentiality cannot be inferred from the available gene essentiality measurements).

6.2 Essentiality predictions

Classifiers trained on both the adjacency features and the ReFeX features showed some skill at predicting reaction essentialities. For a final evaluation of the performance of our models, the best model on each feature set was re-trained on the whole training set and then evaluated on the previously unseen test set.

6.2.1 Standardised Adjacency Matrix

The model with the highest Macro-F1 score on the standardised adjacency features was a Random Forest with hyperparameter settings presented in 5.3.1. Evaluated on the test set, the model had an overall accuracy of 76% and a Macro-F1 score of 67.3%, with a precision of 82.5% and a recall of 86.8% (Figure 6.3).

We note that on the test set, the model has slightly better Macro-F1 score than on average during CV (67.3% vs 64.8%). This is likely caused by the limited size of our test set of 51 reactions but shows that the model performs similar on reactions that did not guide the model selection process.

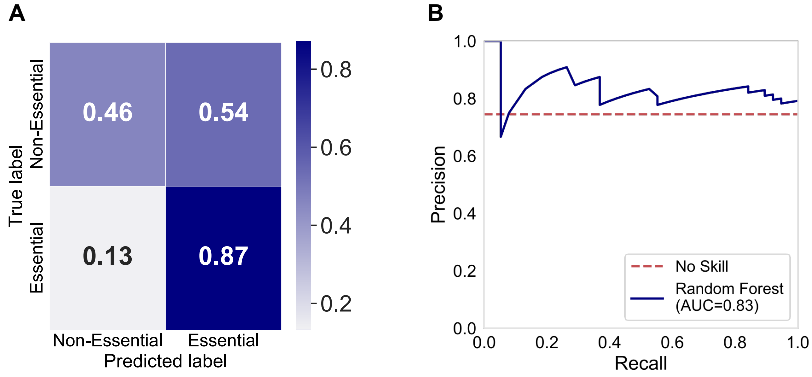


Figure 6.3: Classification performance of the optimised Random Forest model on the reactions in our test set using standardised adjacency features.

6.2.2 ReFeX

Using ReFeX features, the Logistic Regression model had the highest Macro-F1 score after hyperparameter tuning (Section 5.3.2). Again, we trained the model on the complete training set and evaluated its performance on the test set (Figure 6.4A and B). The results show that the model is unable to predict reaction essentiality of reactions in the test set.

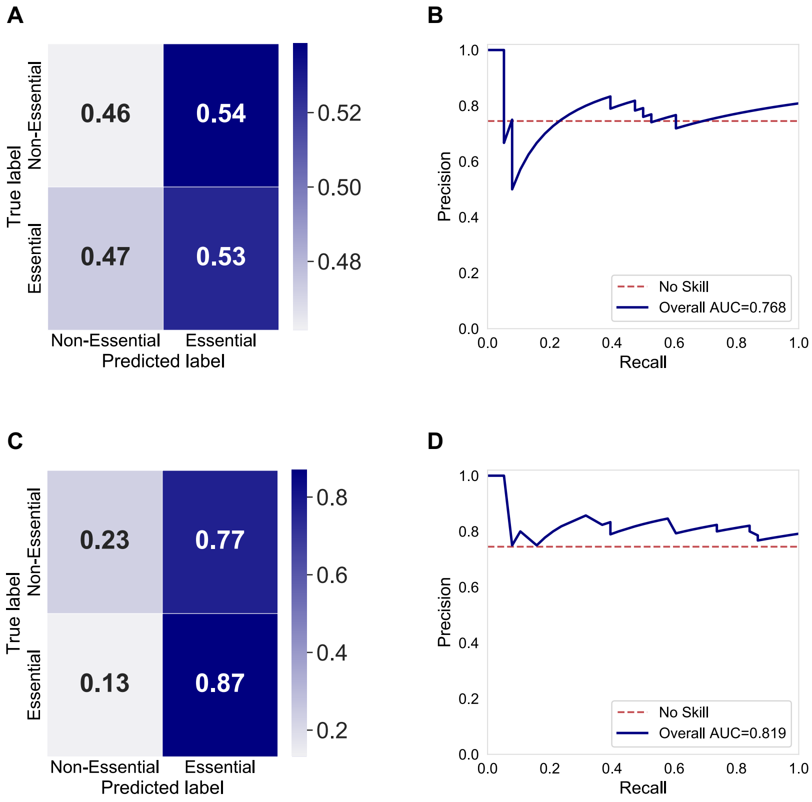


Figure 6.4: Classification performance of the optimised Logistic Regression (A and B) and Random Forest (C and D) models on the reactions in our test set using standardised ReFeX features.

To investigate whether this is only the case for the logistic regression model, we also trained the Random Forest on the complete training set and evaluated it on the test set, motivated by its good test performance using adjacency features. However, we again found that the model is unable to accurately predict reaction essentiality using ReFeX features (Figure 6.4C and D).

6.2.3 Comparison to essentiality predictions of *i*ML1515

From the paper published by Monk et al. (2017), we know that *i*ML1515 has a gene essentiality prediction accuracy of 93.4% across all of their experiments.

However, we are only training and predicting the essentiality of a small subset of reactions in the model, namely the reactions contained in the MFG. To enable a direct comparison of the predictions made by our method and by *i*ML1515, we computed the metrics of *i*ML1515's predictions on the genes associated with the reactions in our test set (Figure 6.5). We find that the *i*ML1515's FBA predictions on these genes have 84.3% accuracy, precision and recall of both 89.5%, with a Macro-F1 score of 79.4%.

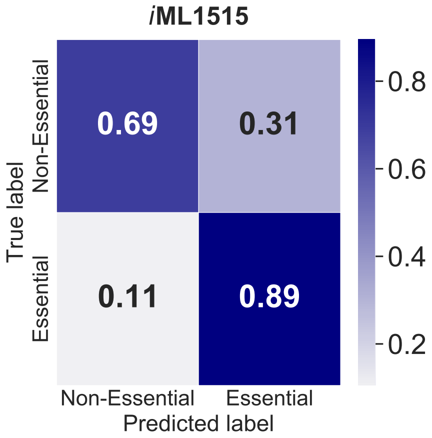


Figure 6.5: Confusion matrix of the FBA gene essentiality predictions of *i*ML1515 on the group of genes in our test set.

Overall, *i*ML1515 shows sub-optimal performance on the test set of reactions, and the whole set of reactions in the MFG, compared to the overall accuracy. This indicates that the set of reactions our classifiers are trained and evaluated on is harder to predict than the average reaction in the metabolic network of *E. coli*.

In the direct comparison with predictions of the Random Forest on standardised adjacency features, we find that our method can predict essential reactions nearly as well as FBA; its TPR is only 2% lower. However, our models struggle noticeably to predict non-essential reactions and only reach a TNR of 46%, compared to *i*ML1515's 69%. Consequently, the Macro-F1 score of our method is 12% lower than that of *i*ML1515.

6.3 Transfer Learning

In our final experiment, we evaluated the Random Forest trained on the glucose MFG on the 252 labelled reactions in the acetate MFG. The model had an overall accuracy of 70.2% and a Macro-F1 score of 64.8%, with a precision of 81.2% and a recall of 76.2%. The corresponding confusion matrix (Figure 6.6A) shows that its true positive rate is 11% lower and its true negative rate is 9% higher than on the test set of reactions of the glucose MFG (Section 6.2.1).

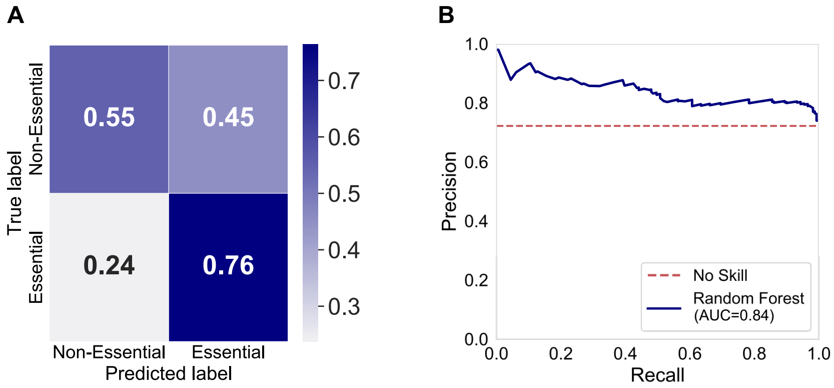


Figure 6.6: Predictions of the Random Forest trained on the MFG of *i*ML1515 using glucose as carbon source on the MFG of *i*ML1515 using acetate as carbon source.

The confusion matrix and PR curve (Figure 6.6B) indicate that the classifier trained on the glucose MFG can predict reaction essentiality in an MFG of a closely related cell.

Chapter 7

Conclusions

In this report, we presented a novel method for essentiality prediction in metabolic networks that combines the currently prominent method, Flux Balance Analysis, with machine learning utilising methods from graph analysis. We used MFGpy, our software package for the generation and mathematical analysis of cell line specific MFGs, to construct and analyse the MFG of the *E. coli* model *i*ML1515. We then applied our proposed machine learning pipeline to this model and showed that simple classification models can predict the essentiality of reactions in MFGs based on network properties.

7.1 Findings

We developed a novel method for predicting gene essentiality, using Flux Balance Analysis in combination with machine learning algorithms. Essentiality prediction was approached as a binary classification problem and reaction essentiality labels were extracted from gene essentiality measurements using an algorithm that we developed.

We then tested our proposed method by applying it to *i*ML1515, the metabolic network of *Escherichia coli*. We simulated cells according to the biological scenarios in which essentiality studies were conducted and verified our simulations by reproducing the FBA gene essentiality predictions of Monk et al. (2017). We used our simulations to generate the MFG of *i*ML1515 and, in combination with reaction essentiality labels inferred from published gene essentiality measurements, created a small dataset of reactions and corresponding essentiality labels.

We created three sets of structural reaction node features from the MFGs. Then, we trained an array of binary classifiers to predict the essentiality labels using these feature sets. The best model was a Random Forest trained on standardised adjacency features with an accuracy of 74.9% and Macro-F1 score of 64.8%. The small number of non-essential reactions in the training dataset posed a challenge to our classifiers and caused low prediction performance on this essentiality class.

Compared to the FBA predictions of *i*ML1515, which were used as benchmark, we found that our method achieves true positive rates near the state-of-the-art, but performs significantly worse at detecting non-essential reactions.

7.2 Answering the Research Question

The aim of this project was to investigate if graphs derived from a cell's wild-type flux distribution contain sufficient information to predict the biological essentiality of metabolic genes. For this purpose, we examined if classification algorithms can predict gene essentiality in *E. coli* using structural features extracted from Mass Flow Graphs derived from the *i*ML1515 *E. coli* model. By creating digraphs from the FBA flux vectors, we were able to approach this problem as binary classification task on graph nodes.

Our classifiers showed promising prediction skills when trained on standardised adjacency features, reaching near state-of-the-art accuracy in their predictions of essential reactions and showing some skill at predicting non-essential reactions. These results indicate that it is indeed possible to predict essentiality based on the wild-type flux distribution only. This is a significant finding as it removes the need to assume that cells aim to maximise growth after specific genes have been deleted; an assumption made when only FBA is used to solve this task.

7.3 Evaluation

A recent review by Aromolaran et al. (2021) identified key needs of machine learning models applied to essentiality prediction: (1) finding relevant features which enable essentiality classification, (2) constructing accurate gold standard class labels for model training, and (3) improving the predictive abilities of models across organisms. We will now evaluate our proposed method on the basis of these key requirements.

We established that structural node features extracted from mass flow graphs contain sufficient information to predict reaction essentiality. By mapping the wild-type flux distribution onto MFGs and treating reactions as graph nodes, we were able to create novel feature sets which, to the best of our knowledge, have not been explored before. Our results indicate their suitability for the problem of essentiality prediction.

By utilising gene essentiality measurements from *in vitro* studies of *Escherichia coli*, we were able to construct biologically accurate gold standard essentiality labels. In contrast to using, e.g., FBA to label the remaining reactions, this ensures the maximum possible reliability of the essentiality labels used for model training. Since we did not employ FBA essentiality values, we removed the requirement to assume optimality of deletion strains; a key objective of this research. However, this posed significant limitations on the size of our dataset; the MFG contains 444 reactions and we can only label reactions that can be deactivated by single-gene knockouts which are 57% of the reactions in the MFG. Consequently, the trained classifiers struggled to correctly predict non-essential reactions; the minority class in our dataset.

So far, we have mainly evaluated essentiality predictions of our models on reactions within the same MFG. Additionally, we briefly investigated if models can predict essentiality in the same cell when grown on a different carbon source. The training and test set therefore originate from the same organism in the same genetic and similar environmental conditions. A key motivation behind exploring the suitability of Flow

Profile and ReFeX features for essentiality prediction was that they can be computed for different MFGs and would potentially be well-suited for transfer learning across more distantly related cells and organisms. Unfortunately, classifiers trained on these feature sets were unable to correctly predict essentiality. Hence, improving the predictive abilities of models across organisms remains an open question for future work, as discussed in the next section.

7.4 Future Work

With the work presented in this report, we answered our research question and fulfilled the main objective of this project. Nonetheless, there are a lot of interesting directions for future work.

Our machine learning models were trained using a very limited amount of data. For further study, additional training data should be generated to train and test our method, e.g. from MFGs of *i*ML1515 corresponding to different growth conditions. We expect that increasing the size of the training dataset will improve the accuracy of essentiality predictions especially for non-essential reactions.

Transfer learning across cells in different environmental and genetic conditions, as well as across organisms should be further explored. Our preliminary results (see Section 6.3) are promising and motivate the application of our method to more MFGs of *i*ML1515 that correspond to different growth conditions. This would enable further investigating the generalisability of the method. It would also facilitate studying conditional essentiality, i.e., genes which are essential only under specific environmental conditions. Next, the analysis should be extended to metabolic networks of other organisms and eukaryotic cells, e.g., cancer cells utilising available gene essentiality measurements (Gatto et al., 2015).

To enable transfer learning across more distantly related cells, further structural feature sets should be explored. As mentioned in 7.3, this would be a key contribution to the field (Aromolaran et al., 2021). Whilst our preliminary transfer learning results are promising, they also underline that finding good features is key. The adjacency features can be used to make predictions in closely related cells such as the same organism in a different environment. However, they are not suited for predictions across organisms as the set of reactions in the MFG has to be similar. A feature set that computes structural features similar to Flow Profiles and ReFeX could potentially overcome this limitation; classifiers trained on these feature sets, however, were not able to accurately predict essentiality. Further studies should therefore be conducted to investigate whether more elaborate machine learning models combined with ReFeX features or Flow Profiles, or different structural feature sets could enable transfer learning across more distantly related cells.

Bibliography

Aromolaran, O., Aromolaran, D., Isewon, I., and Oyelade, J. (2021). Machine learning approach to gene essentiality prediction: a review. *Briefings in bioinformatics*, 22(5). doi:10.1093/bib/bbab128.

Bartha, I., di Iulio, J., Venter, J.C., and Telenti, A. (2018). Human gene essentiality. *Nature Reviews Genetics*, 19(1), 51–62. doi:10.1038/nrg.2017.75. URL `https://doi.org/10.1038/nrg.2017.75`.

Becker, S.A., Feist, A.M., Mo, M.L., Hannum, G., Palsson, B., and Herrgard, M.J. (2007). Quantitative prediction of cellular metabolism with constraint-based models: The COBRA Toolbox. *Nature Protocols*, 2(3), 727–738. doi:10.1038/nprot.2007.99.

Beguerisse-Díaz, M., Bosque, G., Oyarzún, D., Picó, J., and Barahona, M. (2018). Flux-dependent graphs for metabolic networks. *npj Systems Biology and Applications*, 4(1). doi:10.1038/s41540-018-0067-y. URL `http://dx.doi.org/10.1038/s41540-018-0067-y`.

Bergstra, J., Yamins, D., and Cox, D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In S. Dasgupta and D. McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, 115–123. PMLR, Atlanta, Georgia, USA. URL `https://proceedings.mlr.press/v28/bergstra13.html`.

Cardoso, J., Vilaça, P., Soares, S., and Rocha, M. (2012). An algorithm to assemble gene-protein-reaction associations for genome-scale metabolic model reconstruction. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7632 LNBI, 118–128. doi:10.1007/978-3-642-34123-6_11.

Chen, Y. and Xu, D. (2005). Understanding protein dispensability through machine-learning analysis of high-throughput data. *Bioinformatics (Oxford, England)*, 21(5), 575–581. doi:10.1093/bioinformatics/bti058.

Cooper, K. and Barahona, M. (2010). Role-based similarity in directed networks. *arXiv:1012.2726v1*.

Dong, C., Jin, Y.T., Hua, H.L., Wen, Q.F., Luo, S., Zheng, W.X., and Guo, F.B. (2018). Comprehensive review of the identification of essential genes using computational methods: focusing on feature implementation and assessment. *Brief-*

ings in Bioinformatics, 21(1), 171–181. doi:10.1093/bib/bby116. URL `https://doi.org/10.1093/bib/bby116`.

Dreiseitl, S. and Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics*, 35(5), 352–359. doi:https://doi.org/10.1016/S1532-0464(03)00034-0. URL `https://www.sciencedirect.com/science/article/pii/S1532046403000340`.

Ebrahim, A., Lerman, J.A., Palsson, B.O., and Hyduke, D.R. (2013). COBRApy: COnstraints-Based Reconstruction and Analysis for Python. *BMC Systems Biology*, 7(1), 74. doi:10.1186/1752-0509-7-74. URL `https://doi.org/10.1186/1752-0509-7-74`.

Fu, G., Yi, L., and Pan, J. (2018). Tuning model parameters in class-imbalanced learning with precision-recall curve. *Biometrical Journal*, 61. doi:10.1002/bimj.201800148.

Gatto, F., Miess, H., Schulze, A., and Nielsen, J. (2015). Flux balance analysis predicts essential genes in clear cell renal cell carcinoma metabolism. *Scientific reports*, 5, 10738. doi:10.1038/srep10738. URL `https://pubmed.ncbi.nlm.nih.gov/26040780https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4603759/`.

Guo, F.B., Dong, C., Hua, H.L., Liu, S., Luo, H., Zhang, H.W., Jin, Y.T., and Zhang, K.Y. (2017). Accurate prediction of human essential genes using only nucleotide composition and association information. *Bioinformatics (Oxford, England)*, 33(12), 1758–1764. doi:10.1093/bioinformatics/btx055.

Hameri, T., Fengos, G., Ataman, M., Miskovic, L., and Hatzimanikatis, V. (2019). Kinetic models of metabolism that consider alternative steady-state solutions of intracellular fluxes and concentrations. *Metabolic Engineering*, 52, 29–41. doi: https://doi.org/10.1016/j.ymben.2018.10.005. URL `https://www.sciencedirect.com/science/article/pii/S1096717618302933`.

Jolliffe, I.T. and Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374. URL `http://doi.org/10.1098/rsta.2015.0202`.

Kaper, J.B., Nataro, J.P., and Mobley, H.L.T. (2004). Pathogenic Escherichia coli. *Nature Reviews Microbiology*, 2(2), 123–140. doi:10.1038/nrmicro818. URL `https://doi.org/10.1038/nrmicro818`.

Kaslovsky, D. (Accessed 15 Mar 2021). GraphRole: Automatic feature extraction and node role assignment for transfer learning on graphs.
https://github.com/dkaslovsky/GraphRole.git.

Kotsiantis, S.B. (2007). 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*, 3–24. IOS Press.

Larrañaga, P., Calvo, B., Santana, R., Bielza, C., Galdiano, J., Inza, I., Lozano, J.A., Armañanzas, R., Santafé, G., Pérez, A., and Robles, V. (2006). Machine learning in bioinformatics. *Briefings in Bioinformatics*, 7(1), 86–112. doi:10.1093/bib/bbk007. URL `https://doi.org/10.1093/bib/bbk007`.

Lu, Y., Deng, J., Rhodes, J.C., Lu, H., and Lu, L.J. (2014). Predicting essential genes for identifying potential drug targets in Aspergillus fumigatus. *Computational Biology and Chemistry*, 50, 29–40. doi:https://doi.org/10.1016/j.compbiolchem.2014.01.011. URL `https://www.sciencedirect.com/science/article/pii/S1476927114000139`.

Monk, J.M., Lloyd, C.J., Brunk, E., Mih, N., Sastry, A., King, Z., Takeuchi, R., Nomura, W., Zhang, Z., Mori, H., Feist, A.M., and Palsson, B.O. (2017). iML1515, a knowledgebase that computes Escherichia coli traits. *Nature Biotechnology*, 35(10), 904 – 908. doi:https://doi.org/10.1038/nbt.3956.

Montezano, D., Meek, L., Gupta, R., Bermudez, L.E., and Bermudez, J.C.M. (2015). Flux Balance Analysis with Objective Function Defined by Proteomics Data-Metabolism of Mycobacterium tuberculosis Exposed to Mefloquine. *PloS one*, 10(7), e0134014. doi:10.1371/journal.pone.0134014.

Orth, J.D., Thiele, I., and Palsson, B.O. (2010). What is flux balance analysis? *Nature Biotechnology*, 28(3), 245–248. doi:10.1038/nbt.1614.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Plaimas, K., Eils, R., and König, R. (2010). Identifying essential genes in bacterial metabolic networks with machine learning methods. *BMC systems biology*, 4.

Rancati, G., Moffat, J., Typas, A., and Pavelka, N. (2018). Emerging and evolving concepts in gene essentiality. *Nature Reviews Genetics*, 19(1), 34–49. doi:10.1038/nrg.2017.74. URL `https://doi.org/10.1038/nrg.2017.74`.

Sahl, J.W., Morris, C.R., and Rasko, D.A. (2013). Chapter 2 – comparative genomics of pathogenic escherichia coli. In M.S. Donnenberg (ed.), *Escherichia coli (Second Edition)*, 21–43. Academic Press.

Terzer, M., Maynard, N.D., Covert, M.W., and Stelling, J. (2009). Genome-scale metabolic networks. *Wiley interdisciplinary reviews. Systems biology and medicine*, 1(3), 285–297. doi:10.1002/wsbm.37.

Yuan, Y., Xu, Y., Xu, J., Ball, R.L., and Liang, H. (2012). Predicting the lethal phenotype of the knockout mouse by integrating comprehensive genomic data. *Bioinformatics (Oxford, England)*, 28(9), 1246–1252. doi:10.1093/bioinformatics/bts120. URL `https://pubmed.ncbi.nlm.nih.gov/22419784https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3338016/`.

Zampieri, G., Vijayakumar, S., Yaneske, E., and Angione, C. (2019). Machine and deep learning meet genome-scale metabolic modeling. *PLOS Computational Biology*, 15(7), e1007084.

Zheng, A. and Casari, A. (2018). *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. O'Reilly Media, Inc., 1st edition.

Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2), 265–286.

Zou, H. and Xue, L. (2018). A selective overview of sparse principal component analysis. *Proceedings of the IEEE*, 106(8), 1311–1320. doi:10.1109/JPROC.2018.2846588.

Appendix A

Appendix

A.1 Mapping gene essentiality to reaction essentiality

In this section, we present the pseudocode of the algorithm used to find reactions in *i*ML1515 that can be deactivated by single-gene knockouts.

Algorithm 1 FINDREQUIREDGENES(rule)

```
Require: $rule$ not empty
  if $rule$ contains a single $gene$ then
      $genes \leftarrow \{gene\}$
  if $rule$ contains no brackets then
      if $rule$ contains 'or' then $genes \leftarrow \{\}$
      else $genes \leftarrow$ set of all genes in $rule$
  if $rule$ contains brackets then
      $rulebefore \leftarrow$ part of $rule$ that is before the outermost brackets
      $rulewithin \leftarrow$ part of $rule$ that is inside the outermost brackets
      $ruleafter \leftarrow$ part of $rule$ that follows the outermost brackets
      $genes \leftarrow$ FINDREQUIREDGENES ($rulewithin$)
      for $subrule$ in $[rulebefore, ruleafter]$ do
          if $subrule$ is not empty then
              $subgenes \leftarrow$ FINDREQUIREDGENES ($subrule$)
              if the logic operations before the brackets is 'and' then
                  $genes \leftarrow$ union($genes, subgenes$)
              else    // logic operation before bracket is 'or'
                  $genes \leftarrow$ intersection($genes, subgenes$)
  return $genes$
```

The Gene-Protein-Reaction (GPR) rules are provided individually for each reaction in *i*ML1515. In MAPGENETOREACTIONKNOCKOUT, we therefore loop through the rules for each reaction R_j. We use the recursive algorithm FINDREQUIREDGENES to compute the set of genes that, if knocked out individually, would knockout R_j. If the set of genes contains exactly one gene g, reaction R_j is knocked out by the single-gene knockout of g, but not by any other genes. In this case, we assume the essentiality of R_j to equal the essentiality of g following Assumption 2. We therefore append reaction R_j to the list of reactions that are deactivated by the single-gene knockout of g.

Algorithm 2 MAPGENETOREACTIONKNOCKOUTS(*model*)

```
// initialise a dictionary mapping gene knockout to reaction knockout
koReactions ← {model.genes : empty list}
for reaction in model.reactions do
    requiredGenes ← FINDREQUIREDGENES (reaction.gene_reaction_rule)
    if requiredGenes contains exactly one gene gene then
        append reaction to koReactions[gene]
return koReactions
```