ezSTEP: an online platform for training and testing machine learning models for protein expression

Andreas Hiropedi

4th Year Project Report
Artificial Intelligence and Computer Science
School of Informatics
University of Edinburgh

2024

Abstract

The biotechnology industry offers many promising avenues to help against many global challenges. One such avenue is that of "cell factories", where microbes are genetically engineered to produce high-value chemicals, which can be used in the production of pharmaceuticals, food, or fuels from green sources. However, developing cell factories requires many trial-and-error rounds of design and experimental testing, which slows down Research & Development and stifles innovation. To address this issue, several efforts have been made to develop accessible and efficient tools to leverage machine learning models for the task of measuring protein expression directly from the DNA sequence. In this dissertation, we introduce ezSTEP, an online platform designed to meet this need by enabling the training and testing of shallow sequence-to-expression regression models. The ezSTEP platform is publicly available online and can be accessed using the following link: https://ezstep-f617792399bb.herokuapp.com. The results from this dissertation have been reported in the manuscript: *Hiropedi, A., Shen, Y., Oyarzún D. A. "ezSTEP: a web-based tool for Sequence-To-Expression Prediction", 2024.* which will be submitted to a peer-reviewed journal in biotechnology and synthetic biology. The manuscript has been attached to this dissertation in Appendix D.

Research Ethics Approval

This project obtained approval from the Informatics Research Ethics committee.
Ethics application number: 867658
Date when approval was obtained: 2023-12-20
The participants' information sheet, the consent form, and the link to the survey itself are all included in Appendix B.

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.

(*Andreas Hiropedi*)

Acknowledgements

Firstly, I would like to thank my supervisor Diego Oyarzún for his support and guidance throughout the year, and for giving me the opportunity to further my biology knowledge and learn more about the uses of machine learning in this field.

Secondly, I would like to thank Yuxin Shen for her continuous support and guidance with the project, and for her additional insights that helped improve the functionality and usability of our end product.

I would like also to thank all the members of the Biomolecular Control Group for enabling me to have such a great first research experience.

A special mention also goes to all the people who took time out of their busy schedules to help in the usability evaluation of this project and provided great ideas and wonderful feedback used in this research.

Finally, to my loving, supporting family, my amazing girlfriend and my irreplaceable friends for the endless support, encouragement and laughs they've given me over the last four years - this wouldn't have been possible without you.

Contents

Chapter 1

Introduction

1.1 Motivation

Microbial engineering involves the manipulation of microbes in order to develop novel uses, covering a wide variety of applications in the pharmaceutical, food and energy sectors (Roell & Zurbriggen, 2020; Spadiut et al., 2014). Predicting protein expression in the cells from DNA sequences is critical for strain optimization in heterologous protein production (Nikolados & Oyarzún, 2023). However, this has proven to be notoriously challenging, as protein expression involves the complex transcriptional and translational process in biosystems (KIreeva et al., 2018; Shah et al., 2013).

Current advances in high-throughput sequencing and screening techniques have facilitated the acquisition of large sequence-to-expression datasets. Machine learning models can be built on these datasets to deliver protein expression prediction of acceptable accuracy for practical applications. Previous research has demonstrated that convolutional neural networks (CNN) can achieve an R-squared prediction accuracy of approximately 0.9 for ribosome binding sequences (RBS) (Höllerer et al., 2020). Machine learning models have also been trained to predict protein expression from promoter performance (Kotopka & Smolke, 2020b; Penzar et al., 2022). Moreover, deep learning models applied to 5'-UTR sequences can facilitate the in silico evolution of DNA (Cuperus et al., 2017). Nonetheless, such models often suffer from certain limitations when it comes to their usability and interpretability, posing great hurdles and barriers for biology researchers.

Considering these limitations, efforts have been made in order to ease the adoption of machine learning tools to analyze biological datasets. A proposed solution that has proven to be quite promising is automated machine learning (AutoML), which automates the design and deployment of ML pipelines with minimal user intervention (He et al., 2021). Several tools that adopt this AutoML architecture have already been made publicly available. For instance, BioAutoMATED provides users with an end-to-end pipeline for biological sequence design (Valeri et al., 2023). Another example would be iLearnPlus, a web platform for DNA, RNA and protein sequence classification, which wraps together different feature extraction methods and classifiers (Z. Chen et al., 2021). Vaishnav et al. (2022) also created a web platform that enables users to

query a pre-trained transformer model which was trained on 20 million promoter sequences. However, whilst existing platforms tackle the problem of classification, there currently exist no AutoML tools that enable users to train and test their own models for the task of regression.

This dissertation introduces a new platform called ezSTEP (Sequence-To-Expression Predictor). ezSTEP also adopts the AutoML architecture, allowing non-programmers to easily create, train, and evaluate their own sequence-to-expression models in a simple and efficient way. Our platform offers several shallow machine learning models that have been used and tested in current literature (Nikolados et al., 2022), all of which can be customised through a series of selectable input parameters and optional features prior to training and evaluation.

1.2 Research Aims and Objectives

The overarching aim of our research is to provide a user-friendly platform for biologists on sequence-to-expression regression tasks. This aim can be broken down into the following list of objectives:

1. Create a user-friendly platform with a simple, easy-to-use user interface.
2. Ensure the platform supports training and evaluating machine learning models.
3. Make the platform publicly accessible through a web browser, removing the need for local setup.
4. Provide several options for model creation, whilst still maintaining a smooth user experience.

1.3 Dissertation Structure

This dissertation is broken down into six chapters, each detailing important steps of the project:

Chapter 2: This chapter includes a literature review of microbial engineering, protein expression, sequence-to-expression models, and existing tools and their limitations.

Chapter 3: This chapter explores the original concept of our platform and early design before implementation.

Chapter 4: This chapter discusses the implementation details of our platform, in terms of the technologies used, how it was made publicly available, and how we ensure the security and privacy of users' data.

Chapter 5: This chapter focuses on the evaluation techniques used for testing our platform to ensure it aligns with our original objectives.

Chapter 6: This chapter concludes the dissertation by analysing the research findings and results from the evaluation and proposing steps for future work.

Chapter 2

Background

This chapter provides a literature review of the existing methods used for sequence-to-expression modelling. First, we introduce the notion of microbial engineering and how it relates to protein expression, which will help familiarise the reader with the key concepts. We will then focus on sequence-to-expression models, and how machine learning tools can be used to develop such models. Finally, we will discuss how these tools can be integrated into automated pipelines and leveraged by end users, citing three examples from current literature, and then introducing our contribution.

2.1 Microbial Engineering and Protein Expression

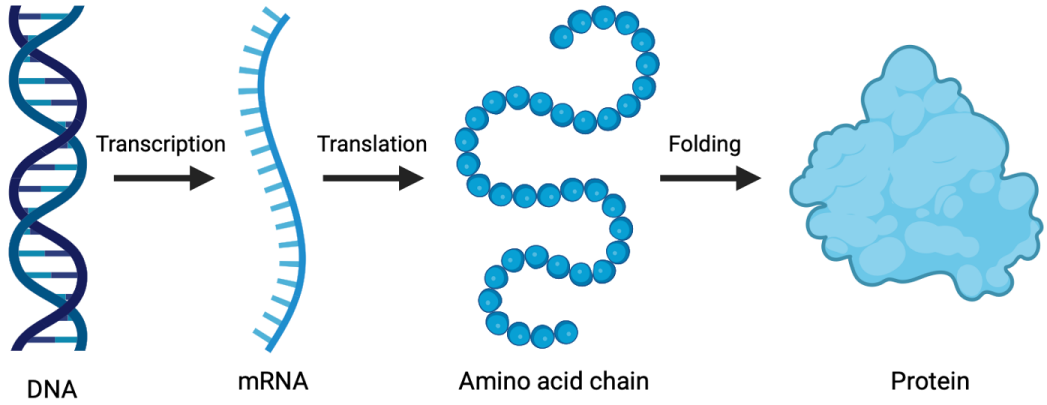


Figure 2.1: The full conversion of a DNA double-helix to a protein.

Microbial engineering involves the manipulation of microbes in order to develop new uses for them, with one such use being the expression of high-value proteins. In order to achieve this, understanding the relation between DNA sequence and protein expression is of the utmost importance: to transcribe a DNA sequence to a protein sequence,

we must first convert the DNA sequence into messenger RNA (mRNA), and then convert the mRNA into protein (see Figure 2.1) (Hopkin et al., 2018).

However, predicting protein expression from DNA sequences has proven to be notoriously challenging, as protein expression involves the complex transcriptional and translational process in biosystems (KIreeva et al., 2018) (Shah et al., 2013). This is heavily impacting microbial strain development, as this process ends up suffering from several costly rounds of design and experimental testing, thus slowing down research and development and stifling innovation.

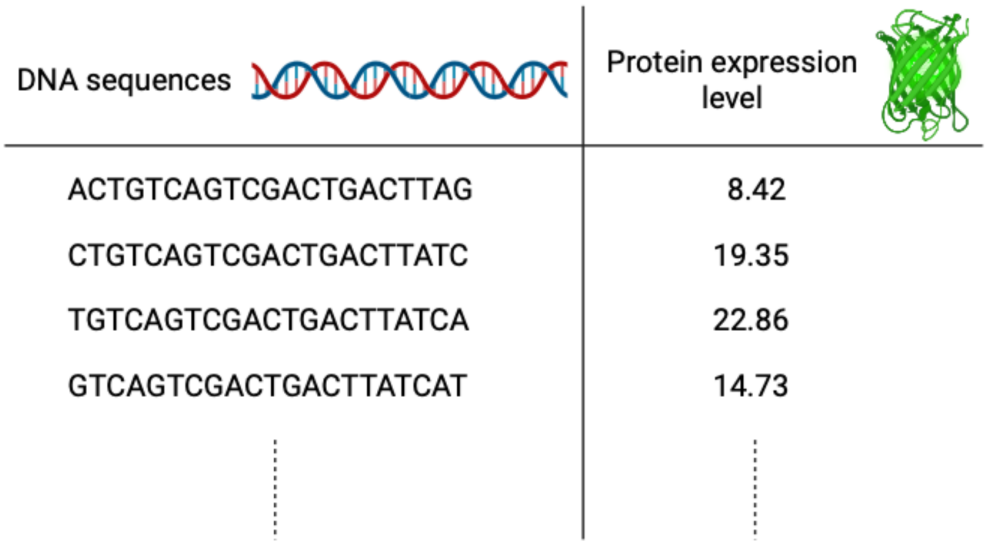


Figure 2.2: Example dataset showing genotype-phenotype associations. Note that fluorescent detectors within the flow cytometer are used to determine the amount of fluorescence emitted by cells, which indicates the level of protein expression (right) (Niedenthal et al., 1996).

Despite these challenges, recent advancements in batch DNA synthesis and high-throughput sequencing have fueled the use of deep mutational scanning to study genotype-phenotype associations. This has resulted in a large number of strategies being developed, all of which have led to the production of really large datasets containing thousands and even millions of genotype-phenotype associations (see Figure 2.2). These large datasets have sparked a growing interest in building *sequence-to-expression* models that can predict protein expression directly from nucleotide sequences, hence foregoing the middle step of transcribing DNA to mRNA before it gets converted to protein (Nikolados et al., 2022).

2.2 Machine Learning for Sequence-to-expression Models

In synthetic biology, the traditional approaches to microbial strain engineering relied on phenotyping libraries of sequence variants and selecting a subset of top producers for further validation, scale-up or iterative design. However, due to the recent advancements and the creation of such large datasets, new model-guided strategies are being developed in order to utilize sequence-to-expression predictors for acquiring knowledge about the phenotypic landscape's structure. We can then repeatedly query such models within sequence optimization loops to explore and navigate the landscape of gene expression (Nikolados & Oyarzún, 2023).

2.2.1 Machine Learning

Among the various strategies for sequence-to-expression modelling, machine learning models have rapidly emerged as a popular and efficient solution to address the problem of accurate prediction for biological sequences (Z. Chen et al., 2021). Recent works have started to incorporate machine learning in the design-build-test cycle for several predictive modelling tasks, including ribosomal binding sequences (RBS) (Höllerer et al., 2020), RNA constructs (Angenent-Mari et al., 2020), promoters (Kotopka & Smolke, 2020a) and other regulatory elements (Cuperus et al., 2017) to name a few.

The growing interest in machine learning tools for sequence-to-expression modelling stems from the availability of large, high-dimensional biological datasets of genotype-phenotype associations. Once trained, these models can be used *in silico* to infer relations between sequence and expression levels (Vaishnav et al., 2022). These relations could further aid researchers in extracting relevant biological insights and help speed up the design of biological sequences with desired properties (Valeri et al., 2023).

2.2.2 Deep Learning

One particular branch of machine learning that has seen great promise for generating accurate sequence predictors is *deep learning*. In essence, deep learning models can be described as data regressors that predict a continuous variable by processing a set of input parameters. One aspect that makes these models stand out is their computational power: they can process incredibly large datasets, deliver highly accurate predictions, and capture complex dependencies with minimal prior assumptions. This, in turn, can help uncover relations in the data on a much greater scale compared to non-deep machine learning models, and one that would be simply infeasible by inspection alone (Nikolados & Oyarzún, 2023).

Deep learning models have enjoyed early successes across several phenotype prediction tasks, including transcription factor binding affinity (Alipanahi et al., 2015), ribosome loading (Sample et al., 2019), and RNA splicing (Jaganathan et al., 2019) to name a few. However, one particularly relevant example from recent literature of the success of deep learning models was the work done by Vaishnav et al. (2022) in investigating the relationship between promoter sequence, expression phenotype, and

fitness. In their research, they generated a large genotype-phenotype dataset with over 20 million randomly generated promoter sequences, as well as their expression levels in Y8205 *Saccharomyces cerevisiae* (*S. cerevisiae*) measured in complex and synthetic media lacking uracil. This dataset was used to train a convolutional neural network (CNN) that can capture fitness landscapes and is able to generalize beyond the training sequence space. The ability to generalize enabled their model predictions to be used as a surrogate for a fitness function in molecular evolution studies (Nikolados & Oyarzún, 2023; Vaishnav et al., 2022).

2.2.3 Limitations of Machine Learning Tools

Despite the wide variety of machine learning models and frameworks that can be used for sequence analysis and prediction, the core underlying mechanisms tend to follow the same generic five-step process: feature extraction, feature analysis, model construction, performance evaluation, and output visualization (Z. Chen et al., 2021). Nonetheless, despite the rather simple outline of their functionality, machine learning tools often pose great hurdles and barriers for biology researchers.

One of these barriers pertains to the trade-off between model accuracy and the budget of experiments. When it comes to deep learning models described in current literature, the vast majority have been trained on fully randomized sequences. While this may seem beneficial, as it prevents models from inheriting any form of bias from the training data, a large number of samples is needed in order to balance the depth and breadth of coverage of the sequence space (Nikolados & Oyarzún, 2023). Additionally, deep learning models such as convolutional neural networks (CNN) require a long training time and computational resources in order to provide a high degree of accuracy (Alzubaidi et al., 2021). Since obtaining such large datasets and computational resources can be very costly, this results in a trade-off between the cost of producing data and the accuracy of the model's predictions. Several considerations regarding the design of training datasets have been discussed in protein engineering literature (Wittmann et al., 2021), and are now emerging within the context of sequence-to-expression models (Nikolados et al., 2022).

Another hurdle comes from the inability to generalize machine learning models due to the design of the training data. In current research, the data needed to train sequence-to-expression models is typically collected using very lab-specific experimental setups and DNA strains. Since these conditions tend to vary significantly across different labs, this results in the inability to extend existing models to predict sequences using unseen data, hence limiting researchers to using their own in-house models (Nikolados & Oyarzún, 2023).

In addition to difficulties faced when it comes to data, there are also concerns when it comes to the actual usability of machine learning models. The sequence-based analysis and predictions tend to require complex processing steps, access to sophisticated software, as well as significant data science expertise (Z. Chen et al., 2021). Moreover, in order to build, train and deploy machine learning models, significant machine learning expertise is also required. Despite the current resources available for scientists, such as online tutorials, open-source code, and a wide range of software packages (Avsec

et al., 2019), there are a large number of user-made decisions that can dramatically influence the quality and performance of machine learning models (Valeri et al., 2023). Such decisions have proven to be rather difficult even among skilled ML practitioners (Zoph et al., 2018), thus raising concerns about the models' usability for researchers with limited ML experience.

2.2.4 AutoML

Considering the limitations outlined in the previous section, several efforts have been made in order to ease the adoption of machine learning tools to analyze biological datasets. A proposed solution that has proven to be quite promising is the use of automated machine learning (AutoML). In essence, AutoML provides methods to automate the design and deployment of ML pipelines with minimal user intervention (He et al., 2021). The techniques provided by AutoML can automatically identify the appropriate model architectures and model hyper-parameters, providing a solid foundation for researchers with limited ML expertise to build initial predictive models (Valeri et al., 2023). There are currently numerous AutoML tools available, spanning from more "shallow" models, such as tree-based optimization using random forest or other scikit-learn methods (Olson & Moore, 2016), to deep learning models such as convolutional neural networks (CNN).

One crucial benefit of AutoML tools, however, is that they can be extended to create automated end-to-end pipelines. Such pipelines would provide scientists with easy data pre-processing, feature extraction, model selection and optimization, and performance visualisation. This would, in essence, address all the current limitations of machine learning tools, by providing researchers with a platform for reaping the benefits of machine learning analysis and prediction in a simple, user-friendly manner (Valeri et al., 2023).

2.3 Previous Work

Given the benefits provided by end-to-end AutoML pipelines, there has been a recent shift towards developing such online platforms to match the needs of end users, namely biology researchers. There are currently several platforms available online; however, in this section, we are only going to focus on three examples.

The three platforms that we are going to consider are iLearnPlus (Z. Chen et al., 2021)), BioAutoMATED (Valeri et al., 2023)), and the Evolution App (Vaishnav et al., 2022)). We have chosen to focus on these three platforms in particular because they address a similar problem to our own. Throughout this section, we are going to critically assess the individual strengths and limitations of each of these platforms.

2.3.1 iLearnPlus

First, we are going to focus on iLearnPlus, which is one of the first tools that automated machine learning for DNA, RNA, and peptide sequences (Z. Chen et al., 2021). iLearnPLus provides the user with the option to download the code and run it on their

own local machine or access the online platform through a website. We assessed the performance usability of the platform by experimenting with several different input parameter combinations.

One thing we instantly noticed is that the user is given plenty of freedom regarding the choice of parameters they can select. Such parameters include feature representation, feature selection, feature normalization are all decisions that are entirely up to the end user. While this may sound good on paper, giving users with limited ML expertise such freedom fails to address one of the current limitations of machine learning models that AutoML tools aim to address, thus taking away some of the usability of the platform.

Another noticeable limitation of the iLearnPlus platform is that, despite the friendly graphical user interface (GUI) (Z. Chen et al., 2021), there is actually very little guidance for the user when it comes to using the platform. Although, before accessing the web server, the user is given a brief explanation of the overall functionality and features available, there is very limited information about the input parameters themselves, which again hinders the usability of this platform for researchers with little ML experience.

Yet another limitation of the iLearnPlus platform is that it is only able to handle classification tasks. This essentially implies that the platform can only handle data with discrete labels, making it impossible for their integrated models to solve tasks involving continuous labels, such as regression problems.

2.3.2 Evolution, Evolvability and Expression App

Next, we will consider the evolution, evolvability and expression app, which serves as a general framework for designing regulatory DNA sequences and addressing fundamental questions in regulatory evolution (Vaishnav et al., 2022). Like iLearnPlus, the user can also choose between downloading the code and running it on their own local machine or accessing the online platform through a website.

Some of the advantages of this platform over iLearnPlus pertain to the usability of the app. For example, one such advantage is that user guidelines are included in order to clarify any ambiguities. Another such advantage is that most of the decision-making process is abstracted away from the user: the only interactions users have with the platform - excluding output visualizations - are for uploading their data and choosing the type of medium between complex and defined media (Vaishnav et al., 2022). These design decisions ensure that less experienced ML researchers can easily interact with the platform, thus rendering it more usable.

However, a huge drawback of this platform is that it only allows the user to query ML models that have already been trained. As a result, this platform only helps with generalising machine learning models to make accurate predictions on different, unseen datasets (Nikolados et al., 2022), but it does not allow users to train such models on their own data and measure their performance.

2.3.3 BioAutoMATED

Lastly, we will shift our focus to BioAutoMATED, an online platform which provides an end-to-end AutoML framework optimized for building models for nucleic acid, peptide, and glycan sequences (Valeri et al., 2023). Unlike the previous two examples, in order to launch this platform the user would need to download the code and run it locally on their own machine.

Like the Evolution app, BioAutoMATED also abstracts most decisions regarding model implementation and design away from the user. However, one added benefit of this platform is that it also enables its users to train machine learning models on the data they uploaded (Valeri et al., 2023). In essence, BioAutoMATED combines some of the main positive aspects of iLearnPlus and the evolution app to provide a powerful tool that is highly usable regardless of the level of ML expertise.

Nonetheless, the BioAutoMATED framework also has its limitations. As previously mentioned, one such limitation is that the platform is not directly accessible through a website, but rather the user needs to set it up on their local device, thus severely hindering its accessibility. Another drawback of the platform is that its attempt to address a wide variety of problems may have come at the expense of usability: there is perhaps too much abstraction from the user, which can potentially result in unclear outputs that are quite difficult to interpret.

2.4 Summary

This chapter presented the current research on sequence-to-expression modelling and identified existing methods and technologies used for measuring protein expression. From the literature, we have seen that AutoML tools serve as a powerful technique that makes sequence-to-expression modelling easier and more accessible to biology researchers with various levels of machine learning expertise. This enables the motivation behind the research to design an online platform that enables users to train and test their own regression models for measuring protein expression. Throughout the following chapters, we will showcase the conceptual design and implementation of the ezSTEP platform, as well as evaluate its performance through a series of experiments using different datasets.

Chapter 3

Concept and Design

In this chapter, we consider the original concept and design of our platform. We start by discussing the AutoML pipeline for our platform, which outlines the main functions of ezSTEP. We then explore the conceptual design of the user interface, bearing in mind how it was created in accordance with good practices for usable software development. Lastly, we focus on how we developed these conceptual ideas into a more concrete set of requirements which can be acted upon and implemented.

3.1 Conceptual AutoML Framework

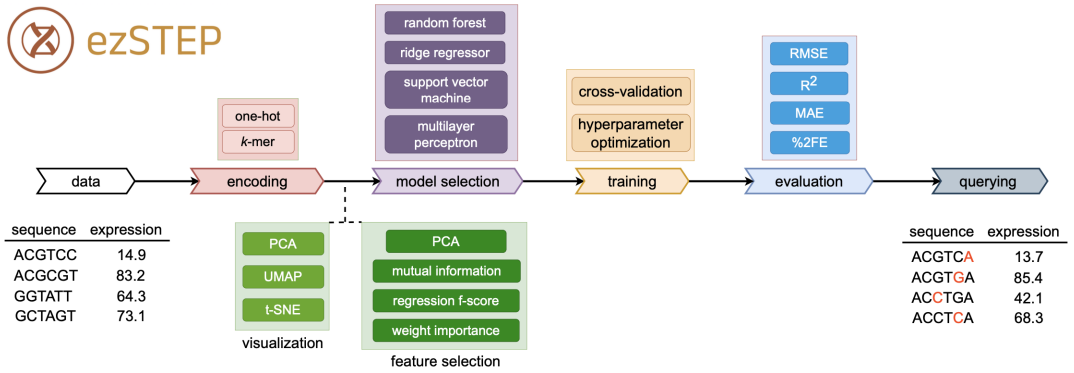


Figure 3.1: The conceptual end-to-end pipeline for training and testing sequence-to-expression models on the ezSTEP platform.

We illustrate the end-to-end pipeline of the ezSTEP platform in Figure 3.1. The DNA sequences initially go through a preprocessing stage, where we convert them to feature vectors and then apply normalisation. These feature vectors are used by the model in order to learn how to accurately estimate the expression levels. After preprocessing, there are two optional steps, namely feature extraction and hyper-parameter optimisation. These steps are only executed if the user enables them as part of their model input selection. The last step before model training is model selection, where the user chooses the type of machine learning algorithm to be used. Once the model has been successfully trained, its performance is then measured on the provided test set as part

of the model evaluation stage. The last step in the pipeline, namely model querying, is optional, and will only be executed if the user supplied a query dataset.

We provide an outline of the available options for each of the steps in our pipeline below:

1. **Feature encoding**: a choice between one-hot encoding or k-mer, with k being any integer value between 2 and 5. One-hot encoding involves converting categorical values, such as DNA sequences, into binary vectors that can be used as inputs to machine learning models (Okada et al., 2019); this technique is the most popular choice for sequence-to-expression modelling (Höllerer et al., 2020). K-mer encoding, on the other hand, involves retrieving all the possible subsequences of length k contained within the DNA sequence and converting them to a vector or matrix form to be used as inputs to machine learning models (Kirk et al., 2018). The k-mer encoding method is very popular in bioinformatics and computational biology.
2. **Feature normalisation**: a choice between the MinMax and Z-score normalisation algorithms. MinMax normalisation involves transforming the range of the feature space such that all values fall within the $[0, 1]$ range (Patro & Sahu, 2015). Z-score normalisation (standardization), on the other hand, scales the data such that the distribution of the normalized values has a mean of 0 and a standard deviation of 1 (Fei et al., 2021). We have chosen MinMax and Z-score normalisation due to their widespread use in data preprocessing for machine learning models (Z. Chen et al., 2021).
3. **Feature selection (optional)**: requires the following two inputs:
   (a) **feature number**: a number between 1 and 100 indicating the number of features to be selected for training.
   (b) **feature selection algorithm**: a choice between PCA (Principal Component Analysis), Weight importance, Regression f-score and Mutual Information. These four algorithms were selected due to their common use in regression tasks (Saeys et al., 2007).
4. **Unsupervised learning (optional)**: a choice between PCA, UMAP (Uniform Manifold Approximation and Projection) or t-SNE (T-distributed stochastic neighbour embedding). These three unsupervised learning techniques were chosen due to their popularity amongst biology researchers, as well as due to the insights they reveal regarding dataset structure (Wong et al., 2016).
5. **Hyper-parameter optimisation (optional)**: uses Bayesian optimisation, requires one input:
   (a) **iteration number**: a number between 1 and 50 indicating the number of iterations that the optimisation is run for.
6. **Model type**: a choice between Random Forest, Multi-layer Perceptron, Support Vector Machine, Ridge Regressor.

For each of our models, we have a series of initial parameters that we set before applying hyper-parameter optimisation, as well as a set search space for certain parameters as part of Bayesian optimisation. These values have been previously tested by Nikolados et al. (2022) and can be seen in Table 3.1 and Table 3.2.

Regressor model	Hyperparameter	Value
Ridge regressor	Regularization (α)	1.0
Random forest	No. of estimators	25
	Maximum depth	30
	Min samples per leaf	3
	Min samples to split	2
Support vector regressor	Kernel method	RBF
	Regularization (C)	30
	Margin tolerance (ε)	0.5
Multi-layer perceptron	Activation function	ReLU
	Hidden layers	3
	No. of neurons	100

Table 3.1: Starting hyper-parameters for non-deep machine learning regressors. These are the parameters set for the models prior to optimisation.

Regressor model	Hyperparameters for optimisation	Search space values
Ridge regressor	Regularization (α)	$[10^{-1}, 10^{2}]$
Random forest	No. of estimators	$[5, 100]$ incr. of 10
	Maximum depth	$[15, 100]$ incr. of 5
	Min samples per leaf	$[1, 12]$
	Min samples to split	$[2, 12]$
Support vector regressor	Regularization (C)	$[1, 50]$
	Margin tolerance (ε)	$[0.1, 2]$
Multi-layer perceptron	Activation function	(ReLU, tanH)
	Hidden layers	$[1, 5]$
	No. of neurons	$[100, 400]$ incr. of 50

Table 3.2: Search space for hyper-parameters for the non-deep machine learning models on the ezSTEP platform. This search space is used for Bayesian optimisation.

3.2 User Interface Design

We illustrate our conceptual design for the user interface in Figure 3.2. As can be seen from the figure, we have opted for a simple, easy-to-follow design, that ensures the desired functionality of the platform. We also wanted to provide the guidelines mostly as a reference point for users in case they get stuck or are unsure of what steps to take.

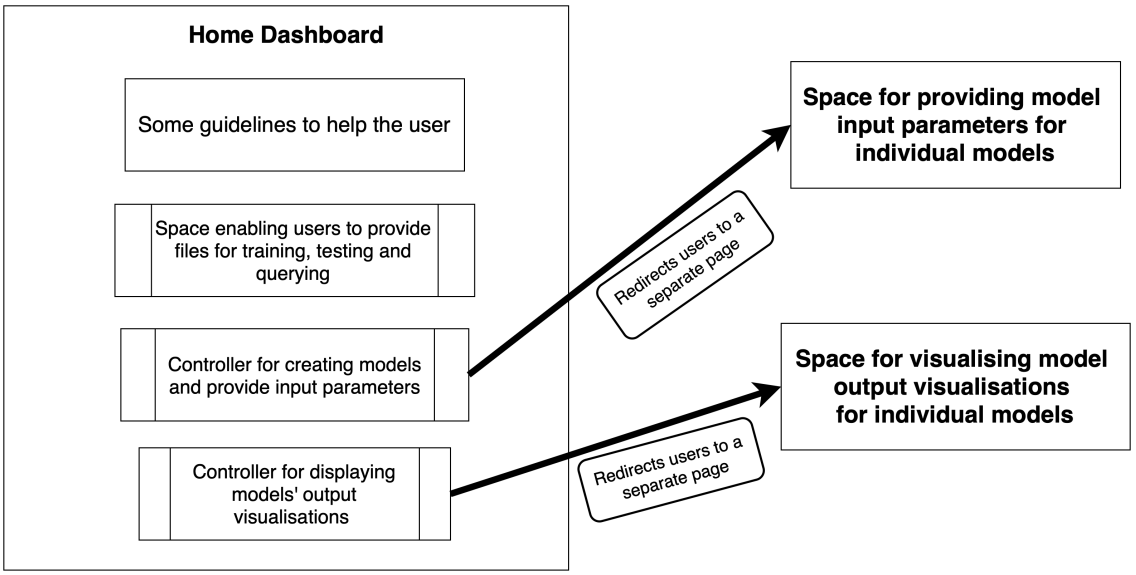


Figure 3.2: The conceptual design of the user interface (UI) for the ezSTEP platform. This is a rough sketch of what would need to be implemented, and provides minimal information on how the UI should look like, and instead focuses on the core functionality of each component. We discuss and show the actual implementation of the UI in detail in chapter 4.

Additionally, since we wanted to enable users to create as many models as they wanted, we concluded that having a separate user interface for providing input parameters for each model would be more beneficial. This is because users can clearly differentiate between the models they create, and it also provides users with the ability to easily go back and alter their parameter selection. Similarly, since we wanted to provide several meaningful visualisations for each individual model, having a separate user interface for the model outputs seemed more optimal.

3.2.1 Nielsen's Usability Heuristics

The conceptual user interface design shown in Figure 3.2 takes into account the principles used in human-computer interaction (HCI) and was designed in accordance with Nielsen's Usability Heuristics. This set of ten heuristics constitutes the most general and widely used principles for designing interactive tools (Nielsen, 1994). We include a list of the 10 principles and their explanation in Appendix A.

An example of how we apply Nielsen's Usability Heuristics to our platform is illustrated through the simplicity of our user interface design and implementation (see Chapter 4). Heuristic 8 states that *Interfaces should not contain information that is irrelevant or rarely needed* (Nielsen, 1994). By ensuring that our design is very minimalist, and the user is neither overwhelmed by the amount of information nor confused by what certain features of the app do, we hope to ensure a high degree of usability whilst still addressing our main objective of providing a platform for training and testing machine learning models.

Another such example would be in heuristic 2, which refers to ensuring a match between the system and the real world. Our design was created considering the following order of events: a user first uploads their data, then proceeds to create a model providing their parameter selection, and then visualises the outputs for that model. We therefore designed our interface in a way that prompts users to perform actions in that exact order but does not limit their freedom to explore the platform's features. For instance, users can freely navigate between the space for uploading files, providing model inputs, and visualising model outputs; however, if a user were to try to create a model without providing any data for those models, the system will inform the user that they will need to go back and upload their data in order to create that model.

3.3 Platform Requirements

Given that the previous two designs were only conceptual, we needed to formalise them in a way that they could be acted upon for implementation. We therefore created a more concrete and actionable set of requirements, which can be broken down into the following three main categories: User Requirements, Task Requirements, and Interface Requirements. We explore each of these categories in more detail in the sub-sections that follow.

3.3.1 User Requirements

ezSTEP's target audience is biology researchers with varying levels of expertise in machine learning (ML). The platform should be accessible and easy to use regardless of this variation in ML knowledge.

3.3.2 Task Requirements

The platform is designed to allow users to customize sequence-to-expression models, train and evaluate them on their own data, and easily interpret and explore the results. This should therefore feature a high degree of customizability, very interactive visualisations, and a simple way that enables users to track the progress of the model creation process (e.g., inform them if the model has been created successfully or if it is still in the training stage). Additionally, there should also be a mechanism that allows users to easily repeat this process for every model they create on the platform.

3.3.3 Interface Requirements

The tool should start with a simple home interface, with a straightforward structure that makes navigating the web-app easy. Additionally, users should be given some information that can further help them with navigating the platform in case they get stuck; nonetheless, this information should not be overwhelmingly large so as to intimidate users instead. Furthermore, when a user is being redirected to a new page, this page should be opened in a new browser tab in order to simplify the navigation process for the user.

Chapter 4

Implementation

This chapter discusses the implementation details of the ezSTEP platform. First, we consider the technologies used as part of implementing the conceptual design described in the previous chapter. We will then discuss how the chosen models were integrated with the platform, and showcase the final implementation of the user interface. Lastly, we will discuss how the web-app was deployed so that it can be publicly accessible, as well as acknowledge the need for security and data privacy in the context of a freely accessible online platform.

4.1 Technology Stack

The technology stack used for the implementation of ezSTEP can be divided into two main parts: models and web platform. This is because the models were implemented separately from the platform, before integrating them together (section 4.2 Model Integration for more details). We discuss the technology stack for each of the two parts in the sub-sections below.

4.1.1 Models

The non-deep machine learning models we provide as part of our platform were built using the scikit-learn Python package (Pedregosa et al., 2011). However, these models were customised with the user inputs provided by the user (such as feature encoding method, feature normalization, etc.), and these were created using custom Python code. We discuss the details of how we integrated the models with the user input, for the purposes of our platform, in the Model Integration section (section 4.2).

4.1.2 Web Platform

For our web platform, we initially considered three options to implement the design outlined in the previous chapter. The three options considered were the streamlit Python package (Richards, 2021), the Plotly Python package (Plotly Technologies Inc., 2015), and the PyQT5 Python package (Meier, 2019). We shortlisted these three

packages due to their widespread adoption amongst developers of AutoML tools. For instance, the iLearnPlus platform was developed using the PyQT5 package (Z. Chen et al., 2021), and the evolution and evolvability app was developed using streamlit (Vaishnav et al., 2022).

However, we decided to use the Plotly package for our platform for two main reasons. Firstly, it provides a high degree of flexibility for customizing features in the user interface through one of its libraries called Plotly Dash (Plotly Technologies Inc., (Plotly Technologies Inc., 2015). The second reason is the ease of integration with the Plotly Graph Objects library, which provides a user-friendly interface with highly interactive graphs that users can easily customize to their needs (Plotly Technologies Inc., 2015). Since creating a highly usable platform was one of the core objectives for this project, the Plotly package seemed like the most suitable option for implementing the ezSTEP platform.

4.2 Model Integration

As we mentioned previously in this chapter, we developed our models using the scikit-learn Python package (Pedregosa et al., 2011). However, this package provides very rigidly defined methods, allowing for minimal model customization. Since our aim is to generate a custom model based on the user's inputs on our interface, we needed to implement an additional mechanism that would allow us to process user inputs and apply them as part of the model creation process.

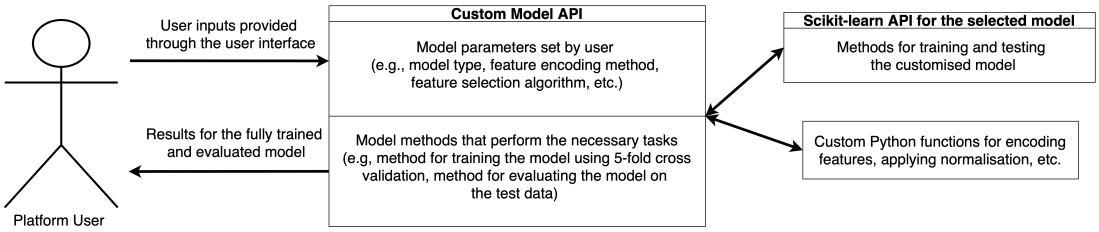


Figure 4.1: Customised model API we created for each of our four models in order to integrate them with our platform. This API records the user's input on the platform, and calls the appropriate methods in order to create, train, and test the user's customised sequence-to-expression model.

The solution we came up with was to create application programming interfaces (API) (Biehl, 2015) for each of the four models we make available on our web-app. Users interact with these APIs simply by providing their inputs through our platform's interface; these inputs are then processed accordingly, and based on the user's model selection, the appropriate API is identified. Once the correct API is found, it then handles calls to the scikit-learn package, ensuring the model is successfully trained and tested on the user's data. However, the API also communicates with some custom Python functions that we implemented. These functions encode each possible input that users can select to customize their model, such as different feature normalisation techniques, or different feature selection algorithms. It is worth noting that these APIs

are fully abstracted away from the user, and they operate entirely as part of the back-end of the ezSTEP platform. The process of how these APIs work is shown in Figure 4.1.

4.3 User Interface Implementation

Given our user interface (UI) design from the previous chapter, we applied a similar split in the implementation of our platform: we divided the user interface implementation into 'home page implementation', 'model inputs page implementation', and 'model output page implementation'. We provide more detail for each of these three components in the sub-sections that follow.

4.3.1 Home Dashboard

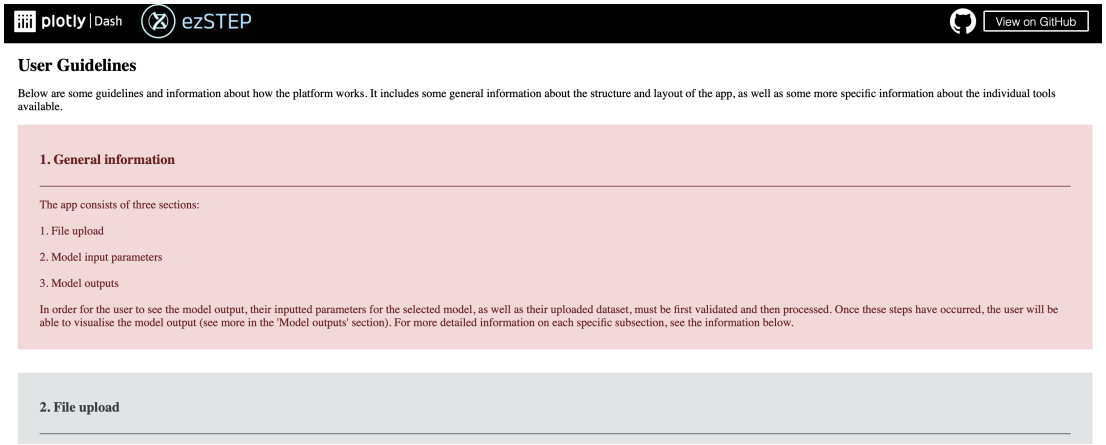


Figure 4.2: Home page view of our platform. This is, in essence, what a user would first see when they access our platform.

As mentioned in Chapter 3, we wanted the home page to act as the main controller of our app. We therefore felt that, when users first launch the app, they should be provided with a set of guidelines that explain the main features of the web-app in very simple terms. In this way, the user is not left guessing what actions need to be taken, which is in accordance with heuristic 10 of Nielsen's Usability Heuristics (see Appendix A) (Nielsen, 1994). We show an image of what users first see when accessing our platform in Figure 4.2.

In addition to the guidelines, the home dashboard also controls the three main aspects of our app: allowing users to upload their data on the platform, enabling users to create models and customize their parameters, and providing users with insightful and informative visualisations of the model's performance on their data. These components are all displayed under individual tabs, which can be found on the home dashboard below the user guidelines. We explore the implementation of each of these individual components in further detail in the sub-sections that follow.

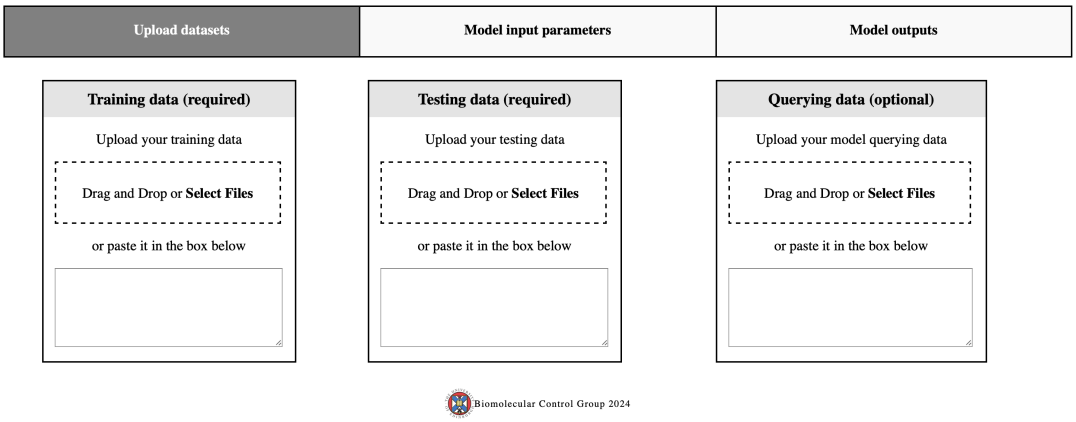
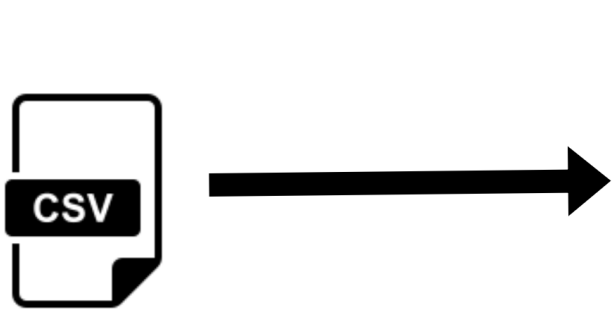


Figure 4.3: The file upload tab on the home page. This is where users can input their own data before creating, training, and testing the sequence-to-expression models on our platform, and they can either use the dropbox provided to upload files or enter the data manually in the textbox below.

4.3.1.1 File Upload

The design of the file upload tab is presented in Figure 4.3. ezSTEP takes training and testing datasets as required inputs, and the querying sequences as an optional input. The datasets can be passed in either as a file, with the accepted format being CSV (see Figure 4.4) or entered manually into the text box shown in Figure 4.3. Once the user provides this data, it is then validated to ensure that it is in the correct format. This includes simple checks such as ensuring correct formatting, as well as checking for irregularities in the data itself, such as DNA sequences not being of equal length or illegal characters being included in the DNA sequences. We also check if the user has provided both a valid file and valid input in the text box, in which case the file will be considered as the final user input to be used for training and testing models.



sequence	protein
ACGAAGCACACACAATT	0.14391
GATGATGTACTATACTC	0.00962
AGCGGTAGTAATACACA	0.07281
ATCTGGGAGAACATGCC	0.65328
ACCGCTGGAAGAGGAAT	0.22439
ACCAACCTACTAAACCA	0.02368
CTGGAGGAAAGATTCTC	0.54375
ATTGTAAGCTTTTGAGG	0.01447

Figure 4.4: Example of the CSV file format used on the ezSTEP platform.

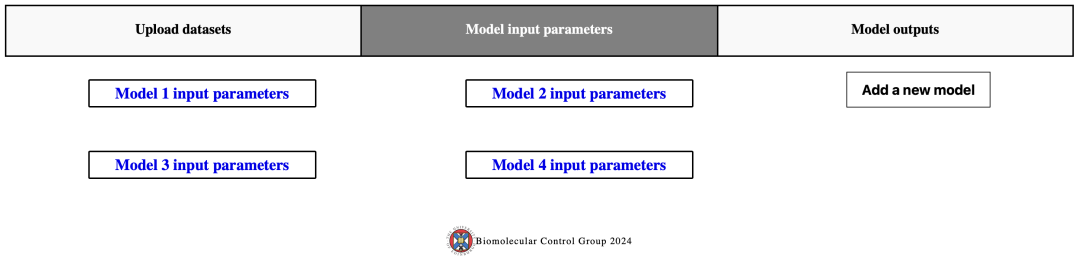


Figure 4.5: The model inputs tab on the home page. Here the users can create as many models as they wish by simply pressing the 'Add a new model' button, and can later customize, train and test them by navigating to the corresponding model page. We provide the hyperlinks to all individual model input pages (blue text that reads 'Model x input parameters') on the platform's home page, as shown above.

4.3.1.2 Model Inputs Tab

We show the user's view of the model inputs tab in Figure 4.5. This tab enables users to create as many models as they wish once they inputted their data. To create a new model instance, users can simply press the 'Add a new model' button (see Figure 4.5).

Since we wanted to provide users the flexibility of creating as many models as possible, and not have to set the parameters of each model one at a time, we decided to create hyperlinks for each model instance a user creates. These hyperlinks are what is actually stored on the home dashboard, and each hyperlink redirects users to that model's corresponding input page, where users can customize that model instance to their liking. We believe this setup provides a very simple and straightforward design for the user, whilst also enabling us to provide more guidance and information on the model creation process through these separate model input pages (see the Model Inputs Page section for more details).

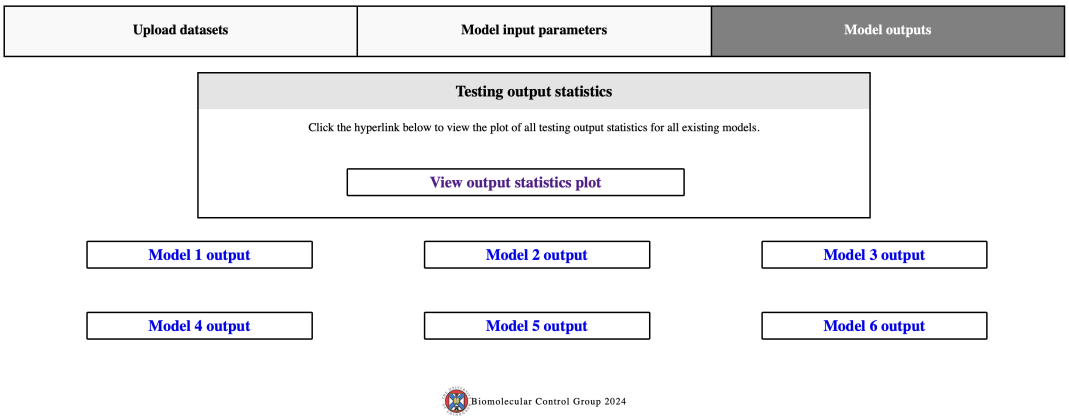


Figure 4.6: The model outputs tab on the home page. This tab can be used for accessing all the output visualisations for each individual model, where users can simply click on the model's hyperlink (blue text that reads 'Model x output') and navigate from the home page to that specific model's output page.

4.3.1.3 Model Outputs Tab

The user interface of the model outputs tab is presented in Figure 4.6. Through this tab, users can access visualisations of the outputs of their models once training and testing have occurred. These plots provide more insight into the performance of the models, as well as their underlying architectures.

Similar to the model inputs tab, the model outputs tab on the home dashboard only stores hyperlinks which redirect users to a specific model's output page (see Figure 4.6). However, we also provide an option that makes comparison across models very easy for the users: by clicking on the 'View output statistics plot' button (see Figure 4.6), users can visualise the performance of all models they successfully created on their provided test data. We believe this setup maximizes the usability of our platform, by providing users with a significant degree of freedom to explore the performance of individual models in depth, as well as compare performance across models in an easy and straightforward manner.

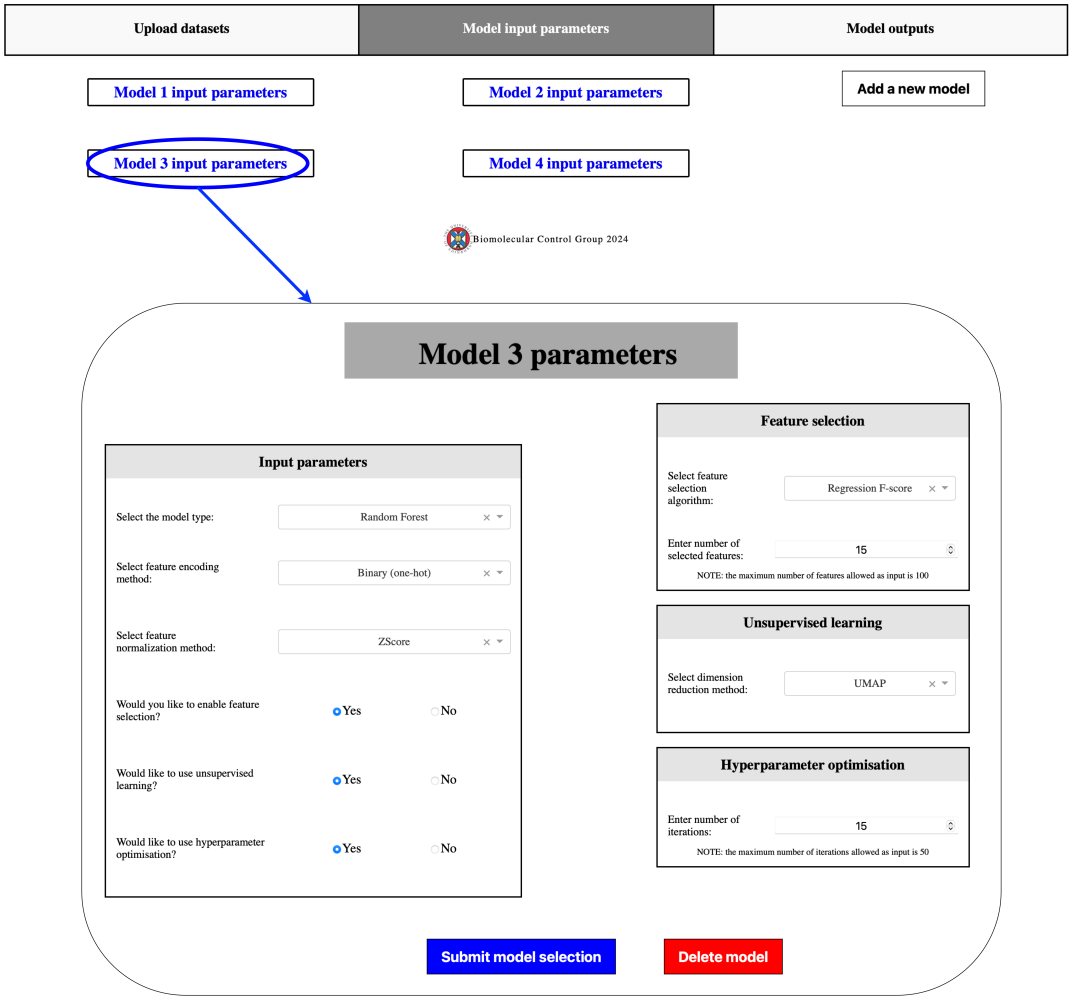


Figure 4.7: The user view of an individual model inputs page. The hyperlinks on the home dashboard (top) redirect users to the respective model's inputs page (bottom), where users can customize the model's input parameters before submitting their selection to create the customized model.

4.3.2 Model Inputs Page

As discussed in the Model Inputs Tab sub-section, the hyperlinks on the home dashboard can be used to access a specific model's input page. These individual pages contain some further guidelines regarding the model creation process and the input parameters available, as well as the actual tools for selecting and submitting a parameter selection for that model. We present the user view of an example model's inputs page in Figure 4.7.

The individual model input pages are responsible for the creation, training, and evaluation of customized sequence-to-expression models. To achieve this, the user first provides inputs for each of the parameters shown in Figure 4.7. Once they have filled in all the fields, they can then submit their selection through the blue 'Submit model selection' button (see in Figure 4.7), which will first ensure that all provided inputs are valid. If all our validation checks pass, the platform then creates the model based on the user's input through the API mechanism described in the Model Integration section (section 4.2) and informs the user once the process has finished. The user is also given an informative message should any of our validation checks fail, which is in accordance with heuristic 5 of Nielsen's Usability Heuristics (see Appendix A) (Nielsen, 1994). This page also provides users with the option to delete models, by simply pressing the red 'Delete model' button (see in Figure 4.7).

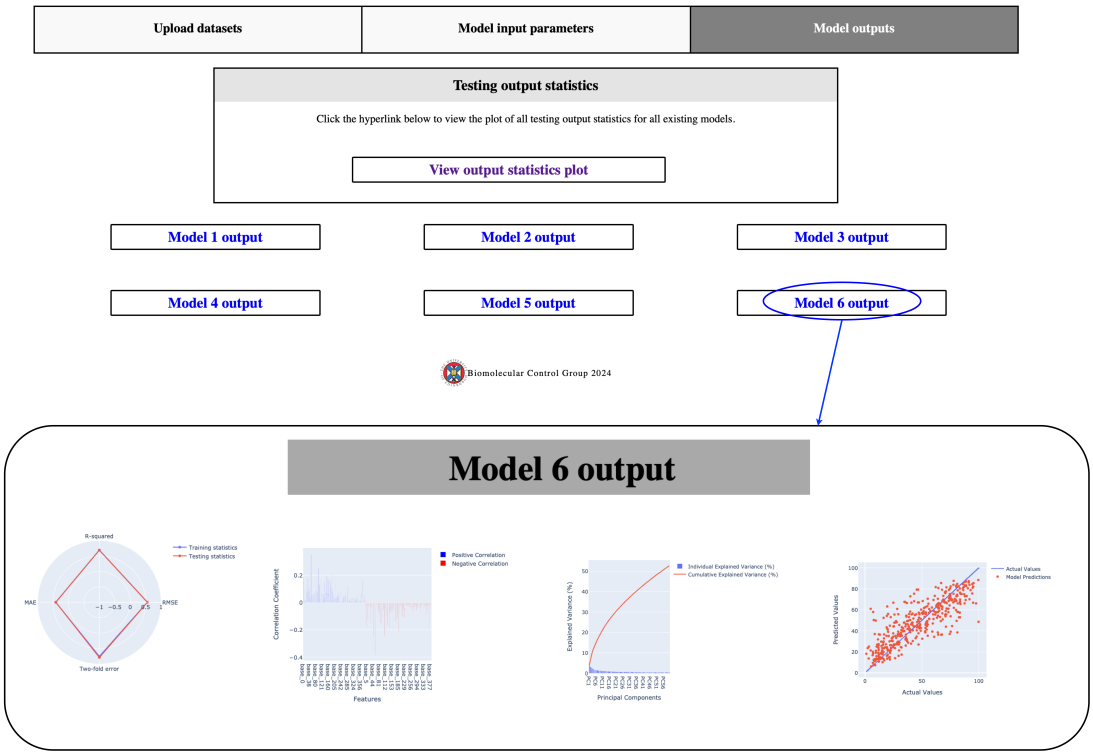


Figure 4.8: The user view of an individual model's outputs page. We show how the hyperlinks on the home dashboard (top) redirect users to that respective model's outputs page (bottom), where users can access several interactive plots about the model's performance and its architecture.

4.3.3 Model Outputs Page

Similar to the previous section, the hyperlinks under the model outputs tab (on the home dashboard) can be used to access a specific model's output page. These pages provide further insights into the model's performance through a series of tables and graph visualisations. We present the user view of an example model's output page in Figure 4.8.

The individual model output pages are responsible for producing all the highly interactive graphs for users to experiment with as part of their exploration and analysis of the results (see Figure 4.8). All these plots are generated using the Plotly package in Python (Plotly Technologies Inc., 2015), and provide several user-interactive features, such as the ability to enable/ disable visualisations for certain components of the graph, zooming in and out, and saving the graph as a PNG file to the user's local device. We provide a short summary of the available graphs on the ezSTEP platform below:

1. A plot showing the training statistics versus testing statistics (for ease of visualisation, since numbers in a table may be harder to interpret).
2. An actual values versus model predictions plot.
3. Plots showing the correlation between different input features (extracted from the DNA sequences) and the target variable (the protein expression) for the training and test data.
4. If the user enables feature selection, a plot showing the explained variance of the selected features.
5. If the user enables unsupervised learning, a PCA, t-SNE or UMAP plot (depending on the user's choice of unsupervised learning algorithm) analysing the provided data.

4.4 Server Deployment

Once the user interface was developed and its functionality tested, we then shifted our attention to making the web-app publicly accessible by deploying it to a web server. We explored several options for deployment, which we outline below:

1. **Amazon Web Services (AWS):** one of the most popular cloud services providers, the main advantage of AWS was its flexibility: offering a system where resources are allocated on the fly based on the platform's requirements (Wittig & Wittig, 2023) meant the scalability of our web-app would no longer be an issue.
2. **Microsoft Azure:** another very popular option for deployment, Azure offers a free account for users to get started, as well as a series of services that are always freely accessible. (Copeland et al., 2015).
3. **Google Cloud:** yet another popular choice, Google Cloud offers its users a free trial period, where users have access to a predefined amount of credits that they can use before needing to pay a fee for the services they are using (Bisong & Bisong, 2019).

4. **Heroku:** perhaps a less known cloud provider, Heroku provides a similar advantage to AWS, albeit with less powerful computational resources for supporting the apps being deployed. Nonetheless, Heroku provides great support for small and medium-scale web apps, and can also very easily be integrated with the Plotly Python package for deployment (Middleton & Schneeman, 2013).
5. **Virtual Machine (VM):** the last approach we considered was using an Ubuntu virtual machine that hosts a web server, and configuring the VM's web server to host our web-app (LaCroix, 2018). The main benefit of this approach would be the ease of management of our app.

The option we have chosen for short-term deployment was Heroku, due to its potential for scalability as well as ease of deployment. This enabled us to meet another core objective of our project, namely that of making our platform publicly accessible. However, in the long term, we aim to shift our app to the Ubuntu Virtual Machine, as this approach allows for easier app management due to the server being in-house. We included further details on the accessibility of our web-app under the Accessibility section in the Conclusion chapter.

4.5 Security and Data Privacy

Given our platform is publicly available, we also needed to implement some precautions to ensure the security and privacy of users' data. To do so, we first assign a session ID to each user who accesses our app. We then map all the information a user inputs to this session ID and store that information in a Redis database as a key-value pair (Carlson, 2013). This ID is unique to each user, and it ensures that the information pertaining to each user is stored separately so as to avoid data leaks.

In addition to unique user IDs, we also ensure that all the packages used as part of our platform are not susceptible to any security vulnerabilities. Since some older package versions can be subject to certain types of security attacks, we have mitigated those risks by using patched versions of these packages that address these security vulnerabilities (Wang et al., 2020).

Lastly, we ensure that the input components on our web-app are not searchable and only accept the expected input values. This additional piece of input validation ensures that any potentially malicious code that could be inserted into our platform through those input components is handled safely and without exposing our users' information to any security risks.

Chapter 5

Testing and Evaluation

This chapter presents the evaluations of the ezSTEP platform, in terms of both the performance of the integrated sequence-to-expression models and the platform's overall usability. We will first consider the metrics used for measuring a model's performance. Afterwards, we will explore how our models perform on three different datasets, each designed for different parts of the DNA sequence. Lastly, we will describe the methodology used to conduct a user study to assess the usability of our platform and discuss the findings of this study.

5.1 Model Performance Evaluation

Our platform measures model performance at both the training and testing stages. We also evaluate the model's performance during training because we employ the technique of 5-fold cross-validation, where we split our training data into sub-sections known as folds (in our case the number of folds is 5), and we reserve one fold for testing and use the remaining 4 folds for training the model (Anguita et al., 2012). This process is repeated five times so that each of the 5 folds is used once to evaluate the model's performance. We opted to use 5-fold cross-validation in the training stage to ensure our model can generalise well to unseen data and avoid overfitting (Moore, 2001).

Once we performed 5-fold cross-validation, we then performed one last training step on the whole training dataset before measuring the model's performance on a separate, unseen test dataset. To evaluate how our models fit the data, we use the same four metrics in both the training and testing stages, and we describe these four metrics in more detail in the sub-section below. It is worth mentioning that, since we obtain five separate values for these metrics as part of the 5-fold cross-validation process, we compute the mean and standard deviation of the obtained values for each metric and report both the mean and standard deviation back to the user (see Figure 5.1).

Evaluation metric	Training value	Testing value
RMSE	0.13 ± 0.004	0.13
R-squared	0.76 ± 0.0148	0.77
MAE	0.09 ± 0.0021	0.09
Percentage within 2-fold error (%)	92.91 ± 1.044	92.31

Figure 5.1: Example of how model evaluation metrics are reported back to the user for both the training and testing stages.

5.1.1 Evaluation Metrics

As mentioned in the section above, we use four key metrics for evaluating the performance of our model. These metrics were inspired by the work done by Hollerer et al. (2020) and were selected for two main reasons. Firstly, these metrics cover a wide range of aspects related to the model's performance, measuring not only whether the model makes correct predictions, but also how close those predictions are to the true value. These insights can be of great help to users, particularly when deciding what models are more reliable for querying. The second reason for choosing these four metrics is because they enable us to accurately compare the performance of our platform's models with that of similar sequence-to-expression models in current literature. The four metrics used (which are also shown in Figure 5.1) are outlined below:

1. **Root Mean Squared Error**: RMSE measures the average difference between values predicted by a model and the actual values using Euclidian distance (Chai & Draxler, 2014). For our models, a lower RMSE would be indicative of good model performance, since the predictions would be very close to the true values. The equation for computing RMSE is provided below (Chai & Draxler, 2014):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

2. **Mean Absolute Error**: MAE is a measure of the average size of the mistakes in a collection of predictions, without taking their direction into account (Willmott & Matsuura, 2005). Similar to RMSE, a lower MAE value would suggest that our model fits the data well, as its predictions are closer to the actual values in the dataset. The equation for computing MAE is provided below (Willmott & Matsuura, 2005):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

3. **Coefficient of determination** (R^2): this is a statistical measure that indicates how much of the variation of a dependent variable is explained by an independent variable for a regression task (Miles, 2005). In our case, a higher R-squared (closer to 1.0) is desired, as it would suggest that the model is a better fit to the

data. The equation for computing R-squared is provided below (Miles, 2005):

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

4. **Percentage within 2-fold error**: this metric refers to the proportion of predictions that are within a 2-fold range of the observed values, which is to say within 2 standard deviations of the true values (Sinha et al., 2008). In the case of our models, a higher value would indicate that the model's predictions are highly accurate since they are very close to the actual values in the original data. The equation for computing the percentage within 2-fold error is provided below (Sinha et al., 2008):

$$\textit{Percentage within 2-fold error} = \left(\frac{1}{n}\sum_{i=1}^{n} 1\left[\frac{1}{2} \leq \frac{\hat{y}_i}{y_i} \leq 2\right]\right) \times 100$$

5.2 Testing on Coding Sequences

The first dataset used for assessing the performance of our models originated from the work done by Cambray et al. (2018). This dataset was the original basis for the work done by Nikolados et al. (2022) on the models integrated with our platform. Therefore, we chose this dataset to ensure our results align with those of Nikolados et al. (2022). In the following sections, we provide a detailed description of this original dataset, describe any preprocessing that was applied to the original data, and report the performance of our models.

5.2.1 Dataset Description and Preprocessing

The Cambray dataset consists of fluorescence measurements for an sfGFP-coding sequence in *Escherichia coli (E. coli)*, fused with more than 240,000 upstream 96 nucleotide (nt) variants that were designed to perturb translational efficiency and the resulting expression level (Nikolados et al., 2022). These 96nt sequences were designed from 56 seeds with maximal pairwise Hamming distances, where each seed was subject to controlled randomization using the D-Tailor framework, so as to produce mutational series with controlled coverage of eight biophysical properties at various levels of granularity: nucleotide sequence, codon sequence, amino acid sequence, and secondary mRNA structure (Cambray et al., 2018; Nikolados et al., 2022). We show a detailed description of this dataset in Figure 5.2.

Since the size of the original data (244,000 sequences) was too large, and the data was split into 56 mutational series containing roughly 4000 sequences each, we decided to use just one of those mutational series to assess the performance of our models. The mutational series was chosen at random, and we then applied a 90/10 split to create our training and testing datasets. The structure of the obtained training set is shown in Figure 5.2B, and we discuss how our models performed on this dataset in the next sub-section.

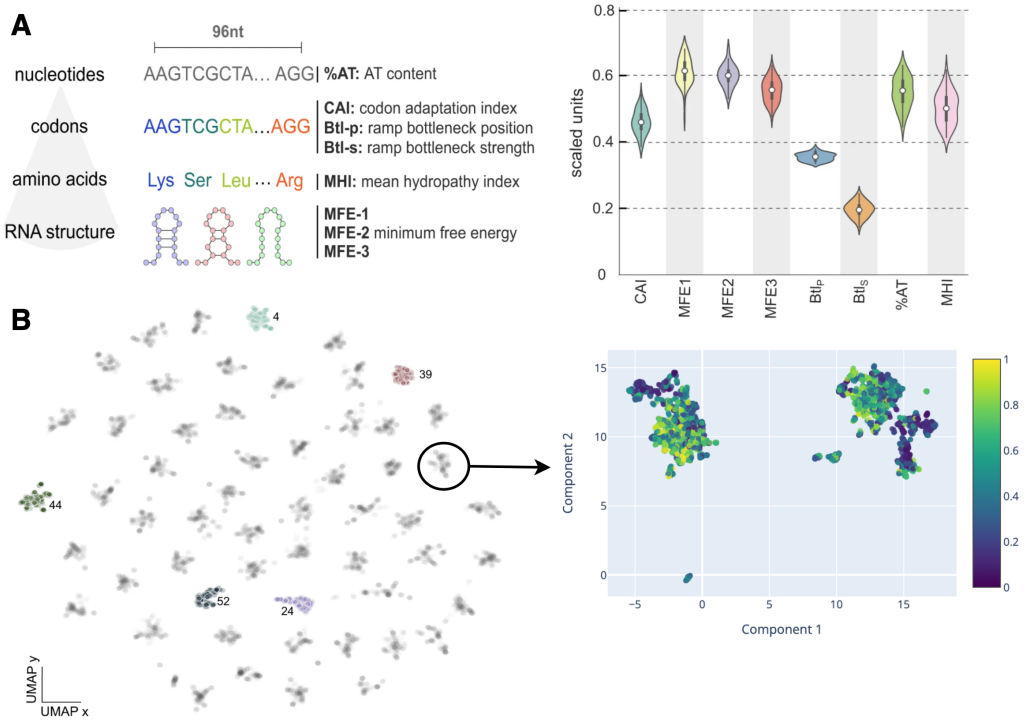


Figure 5.2: A A large phenotypic screen in Escherichia coli of an sfGFP coding gene preceded by a variable 96nt sequence. The variable region was designed on the basis of eight sequence properties that impact translational efficiency. Violin plots show the distribution of the average value of these eight properties across the 56 mutational series in the dataset (figure obtained from the paper by Nikolados et al. (2022)) **B** Two-dimensional UMAP visualization (left) of overlapping 4-mers computed for all sequences in the dataset, with each cluster corresponding to a mutational series (figure obtained from the paper by Nikolados et al. (2022)). We then zoomed in on one of these mutational series which we used as a training dataset for our models (plot on the right), and we created a two-dimensional UMAP visualisation of overlapping 4-mers for all the sequences in that specific mutational series.

5.2.2 Observed Model Performance

The performance of our platform's models on the Cambray dataset is shown in Figure 5.3. Our results show that, for datasets of around 4000 sequences, fairly good performance ($R^2 \geq 0.70$) can be obtained using most of the models on our platform. We observe that the linear ridge model performs exceptionally poorly ($R^2 = 0.48$), whereas the support vector regressor (SVR) and the two non-linear models perform much better. We found random forest regressors to be the most accurate among the considered models, with a significantly higher R^2 value ($R^2 = 0.77$). This could be due to the model's architecture: random forest uses decision tree ensembles, which are more powerful and capable of capturing non-linear relationships between features and the target variable (Rodriguez-Galiano et al., 2015).

Another observation that we noted is the impact of DNA encodings as well as that of feature normalisation methods on prediction accuracy. We found that, between the

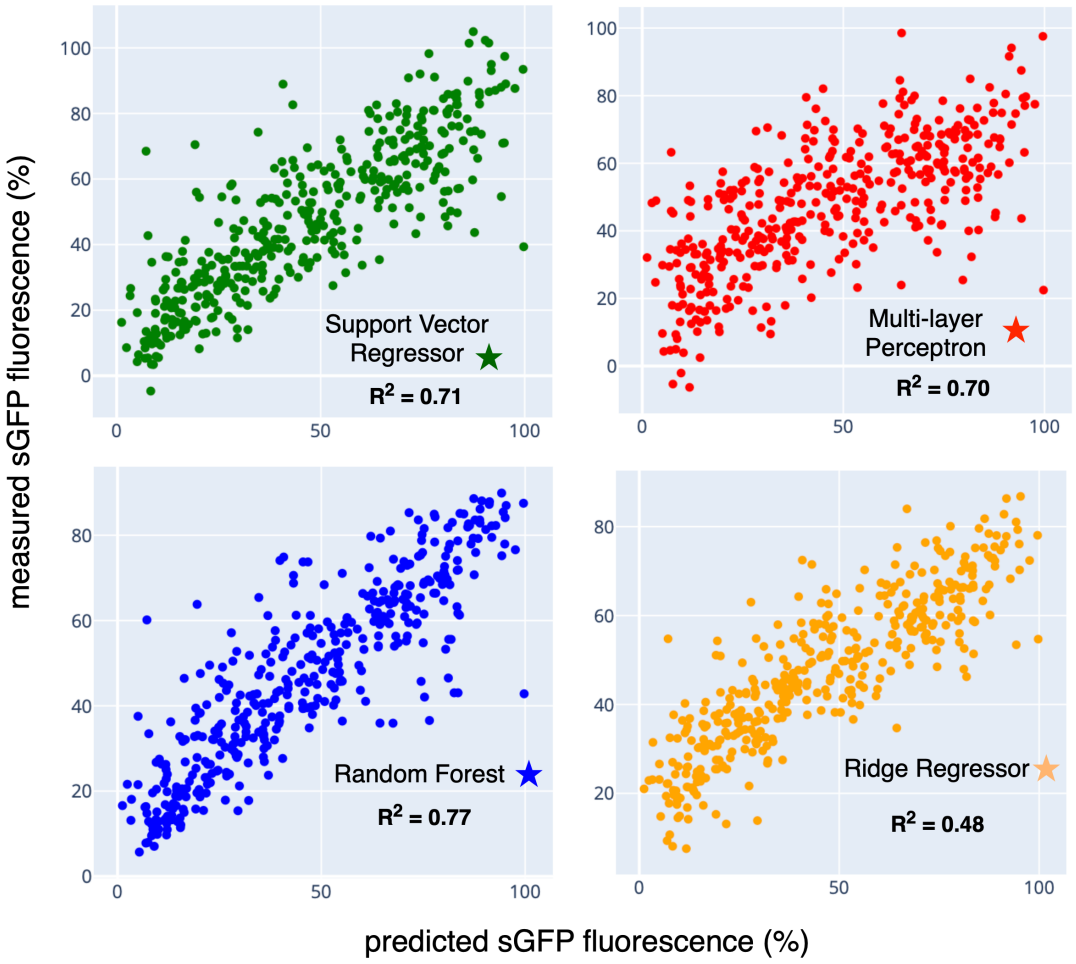


Figure 5.3: Example scatter plots of model predictions on the Cambray test dataset for our four models (marked with stars), together with their R-squared value for performance. These scatter plots were obtained directly from the ezSTEP platform.

feature encoding methods available on our platform, one-hot encoding offers the best performance, with the difference being as staggering as a 0.05 decrease in R^2 value compared to some k-mer encodings such as 2-mer. Additionally, for models such as SVR, which are highly sensitive to feature scaling (Pisner & Schnyer, 2020), Z-score normalisation achieved overwhelmingly better performance on the base model (i.e., without applying hyper-parameter optimisation): the R^2 value of 0.71 shown in Figure 5.3 was obtained by applying Z-score normalisation, whereas MinMax normalisation produced a negative R^2 value for the SVR model.

Lastly, it is worth mentioning that, by applying Bayesian optimisation for our models' hyper-parameters, we note a significant performance increase, with even linear models such as the ridge regressor achieving R^2 values of around 0.70, whereas better-performing models such as random forest achieve R^2 values of $R^2 \geq 0.80$. These findings generally align with those from the paper by Nikolados et. al (2022), and at times even surpass the paper's findings when hyper-parameter optimisation is used.

5.3 Testing on Additional Datasets

In addition to the coding sequences dataset developed by Cambray et al. (2018), we have further assessed the performance of our models on two additional datasets. We explore the specifics of these datasets, as well as the reasons for selecting them, in the sub-sections that follow.

5.3.1 Testing on Promoter Sequences

We first shifted our attention from the 5'-end of the coding sequences to natural yeast promoter sequences. The dataset is a nature promoter dataset, which was originally used as a validation set for the model used by Vaishnav et al. (2022) for their app. We therefore felt this dataset would be perfect for measuring the performance of our models on a different part of DNA, as there were already existing results in current literature (Nikolados et al., 2022) that we could use for comparison. In the following sub-sections, we provide a detailed description of this additional dataset, describe any preprocessing that was applied to the original data, and report the performance of our models.

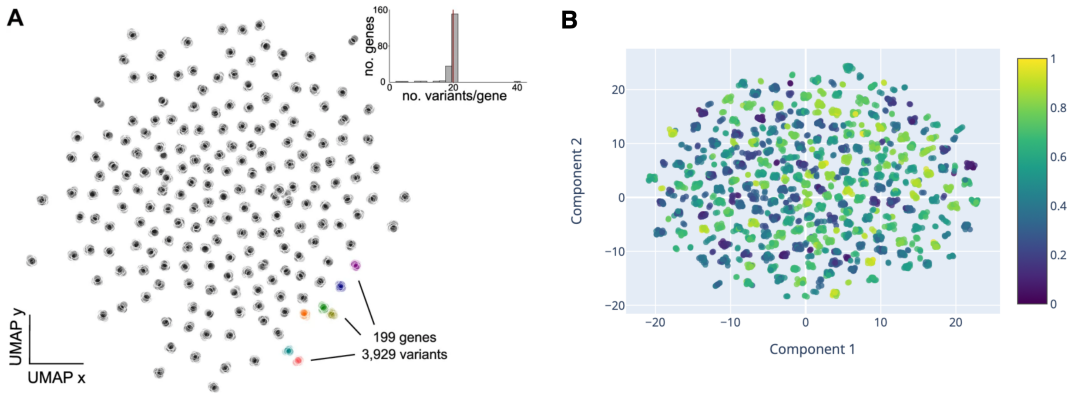


Figure 5.4: A Genotypic space of yeast promoter data from Vaishnav et al. (2022) visualized using a 2D UMAP plot; sequences were featurized using counts of overlapping 4-mers. The dataset contains 3929 promoter variants (80nt long) of 199 native genes, as well as fluorescence measurements of a yellow fluorescent protein (YFP) reporter; the inset shows the distribution of variants per gene across the whole dataset (figure obtained from the paper by Nikolados et al. (2022)). **B** Two-dimensional UMAP visualisation for the training data we obtained following our 90/10 split on the original dataset; sequences were once again featurized using counts of overlapping 4-mers.

5.3.2 Dataset Description and Preprocessing

The original dataset from the Vaishnav et al. (2022) papers consisted of more than 30 million sequences in complex medium (yeast extract, peptone and dextrose (YPD) and more than 20 million sequences in defined medium (synthetic defined medium lacking uracil (SD-Ura)) (Vaishnav et al., 2022). These sequences 80 nucleotide (nt) long sequence of promoter DNA from *Saccharomyces cerevisiae (S. cerevisiae)*, more

commonly known as yeast. However, the paper also introduces a dataset using the same type of DNA sequence (80nt promoter sequences), but which only comprises of around 4000 sequences.

We therefore opted for this smaller validation set as it was not only a better fit for evaluating the models on our platform (since the original data was too large) but there also exists previous work done by Nikolados et al. (2022), where they measure the performance of non-deep machine learning models on this data. This dataset therefore also provided us with a benchmark to ensure that our results align with findings in current literature.

The structure of this dataset is presented in Figure 5.4. The data is comparable to that in Cambray et al. (2018) in both sequence length (80nt compared to 96nt in the Cambray data) as well as highly clustered coverage of genotypic space (Figure 5.4A). This clustered structure results from the design of the data itself, which is composed of 3929 variants of 199 natural promoters. A key difference between this dataset and Cambray et al. (2018) is the construct architecture: unlike the UTR sequences shown in Figure 5.2B, promoter sequences account for regulatory effects but do not undergo transcription (Nikolados et al., 2022). Similar to the Cambray dataset, we applied a 90/10 split to create our training and testing datasets, and we show the structure of the obtained training set in Figure 5.4B. We discuss how our models performed on this new dataset in the next sub-section.

5.3.3 Observed Model Performance

The performance of our platform's models on the yeast dataset is shown in Figure 5.5. The results for this dataset are significantly higher than those for the coding sequences dataset (Cambray et al., 2018), with all models achieving R^2 values of $R^2 \geq 0.80$, and even the linear ridge regressor achieving a high R^2 of 0.85 before applying optimisation. We also see that the non-linear models (random forest and multi-layer perceptron) achieve values as high as $R^2 \geq 0.90$, with results getting as high as $R^2 = 0.96$ for random forest with Bayesian hyper-parameter optimisation. We also see that, unlike the Cambray dataset, DNA sequence encoding methods and normalisation algorithms have a much more limited impact on the performance of the models, with a difference of only 0.01 in R^2 value.

Although these results are highly encouraging of the models' ability to generalise well to different parts of the DNA sequence, we can partly attribute the measured model performance to the structure of the dataset. As we have seen in Figure 5.4 and discussed in the previous sub-section, the yeast dataset is highly clustered, meaning that the feature space for this dataset is highly correlated. This can also explain why the non-linear models outperform the linear models on this data since their architecture allows them to exploit these correlations amongst features to more accurately predict the target variable (Rodriguez-Galiano et al., 2015).

Lastly, we note that, similar our results for the Cambray dataset, these findings also align with those from the paper by Nikolados et. al (2022), and even surpass the paper's findings when hyper-parameter optimisation is used.

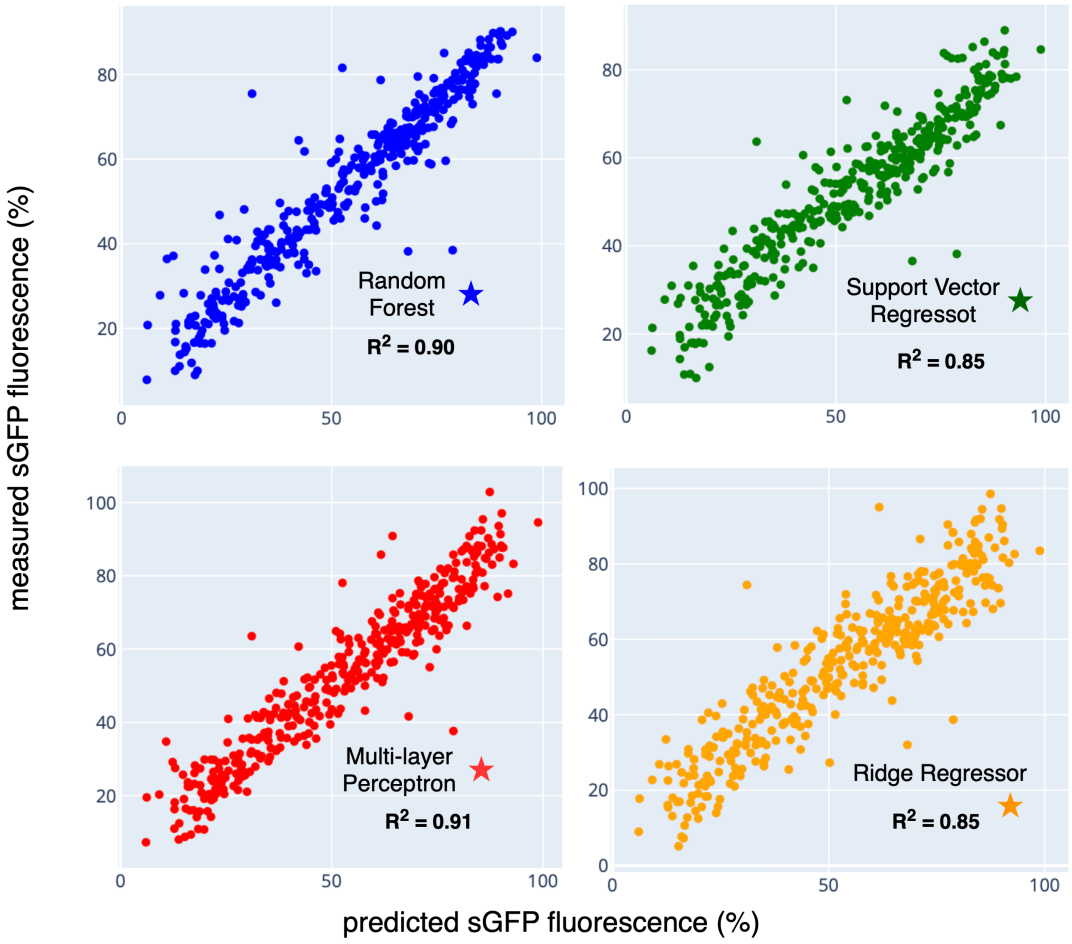


Figure 5.5: Example scatter plots of model predictions on the yeast test dataset for our four models (marked with stars), together with their R-squared value for performance. These scatter plots were obtained directly from the ezSTEP platform.

5.3.4 Testing on Ribosome Binding Site (RBS) Sequences

The second additional dataset is on the ribosome binding sites (RBS) originated from the work done by Hollerer et al. (2020). The structure of this dataset is vastly different than that of the promoter sequences dataset (Vaishnav et al., 2022), making it ideal for ensuring that our models can generalise well to different parts of the DNA sequence. Moreover, since the work done by Hollerer et al. (2020) also provides results for some of the models on our platform, we can compare the performance of ezSTEP's models with already existing results from literature. In the following sub-sections, we provide a detailed description of this additional dataset, describe any preprocessing that was applied to the original data, and report the performance of our models.

5.3.5 Dataset Description and Preprocessing

The original dataset from the Höllerer et al. (2020) paper consists of around 300,000 bacterial ribosome binding site (RBS) sequences (Höllerer et al., 2020). Similar to the Cambray et al. (2018) data, these sequences were also obtained from *Escherichia coli (E. coli)*; however, the RBS sequences used are much shorter, with a length of only 17 nucleotides (17nt) compared to the 96nt coding sequences in the Cambray dataset (Höllerer et al., 2020). The structure of this dataset is presented in Figure 5.6.

Since the size of the original data (303,503 sequences) was too large, we needed to apply some preprocessing in order to use it for model evaluation. We therefore randomly sampled around 4000 sequences from the original data, and then performed a 90/10 split on the sampled sequences to create our training and test datasets. The structure of our training data is shown in Figure 5.6C, and we discuss how our models performed on this dataset in the next sub-section.

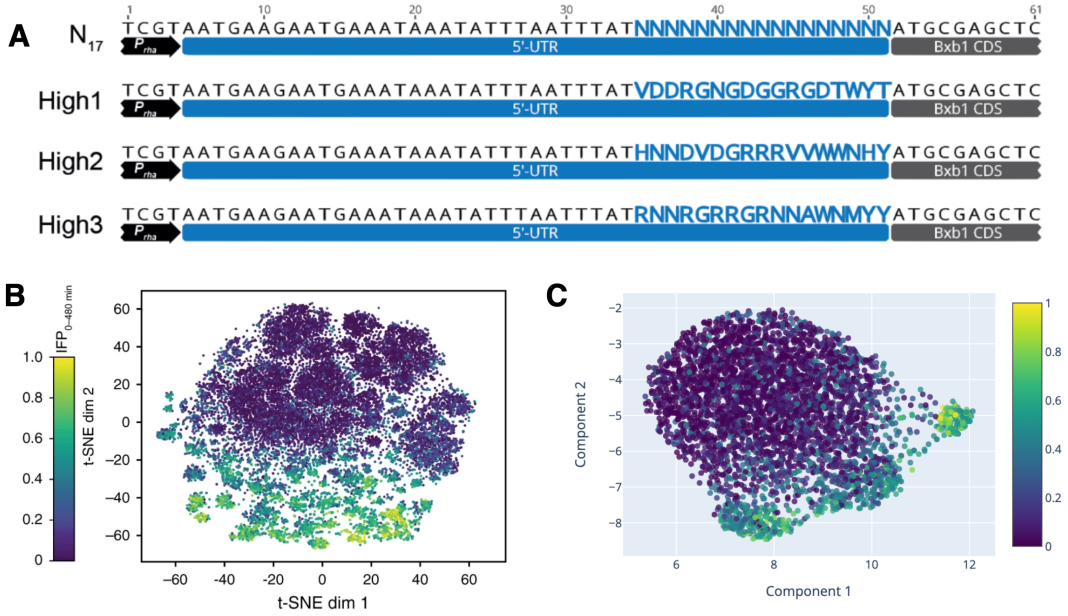


Figure 5.6: **A** Ribosome binding sites (RBS) libraries constructed in the Hollerer et al. (2020) paper. 17 consecutive base pairs upstream of the bxb1 start codon were randomized. A fully degenerate library (N17) as well as three libraries with 84 reduced skew towards weak RBS (High1, High2, High3) were constructed (figure obtained from the paper by Hollerer et al. (2020)). **B** Two-dimensional T-distributed stochastic neighbour embedding (t-SNE) of the original dataset in the Hollerer et al. (2020) paper (figure obtained from the paper by Hollerer et al. (2020)). **C** Two-dimensional UMAP visualisation for the training data we obtained following our 90/10 split on the original dataset.

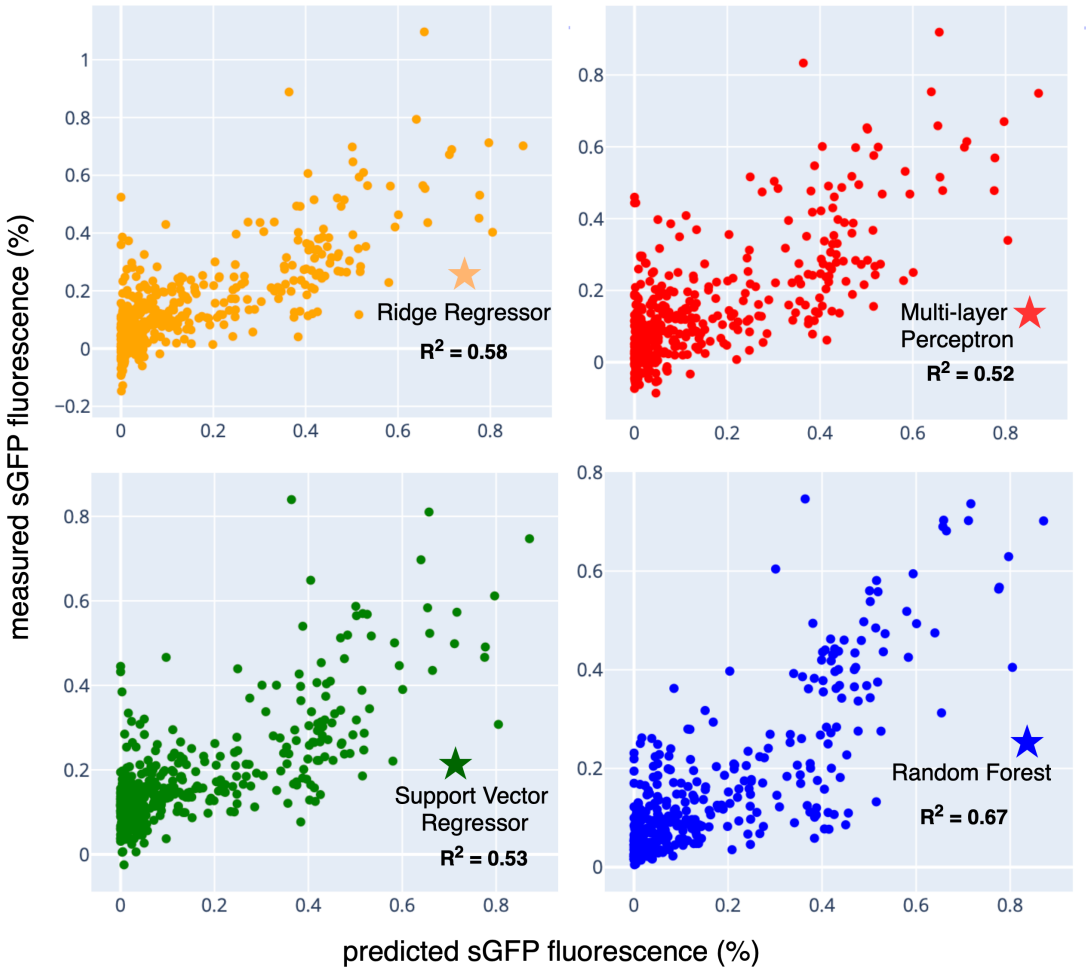


Figure 5.7: Example scatter plots of model predictions on the Hollerer test dataset for our four models (marked with stars), together with their R-squared value for performance. These scatter plots were obtained directly from the ezSTEP platform.

5.3.6 Observed Model Performance

The performance of our platform's models on the Hollerer dataset is shown in Figure 5.7. The results for this dataset are significantly lower than for the other two datasets but are nonetheless encouraging, with an R^2 value of $R^2 \geq 0.50$ for all models. Unsurprisingly, the model that performed best on the previous two datasets, namely random forest, also achieved the highest R^2 value on the RBS data ($R^2 = 0.67$). This appears to suggest that decision tree-based models can generalise well across different parts of the DNA sequence. Additionally, our observed performance for the random forest model also aligns with that in the Höllerer et al. (2020) paper.

Interestingly, perhaps, is that both linear models (ridge regressor and SVR) outperformed the multi-layer perceptron (MLP) model on this dataset (see Figure 5.7). This could, however, be explained by the structure of the data: relative to the other datasets, this data contains a lot more noise due to the random sampling from a large space of sequences (randomly sampling 4000 sequences from over 300,000 sequences). Since

MLP models are less robust to outliers compared to linear models (Timpl et al., 2022), this makes them more prone to making less accurate predictions.

5.4 Platform Usability Testing

In addition to evaluating our models' performance on a wide variety of datasets, we also wanted to assess our platform's overall usability. To effectively measure usability, we opted for a user study with human participants. We describe the methodology used when conducting this study, as well as its results, in the next two sections.

5.4.1 Methodology

The main aim of our study was to prove that the web-app is highly user-friendly, particularly for novice users. We hoped to show that the simple user interface design provides clarity, the interactive features of our platform (such as interactive graphs) provide informative insights, and the process of using ezSTEP is efficient and enjoyable.

In order to test this appropriately, we have asked users to indicate their level of expertise in two separate disciplines, namely programming and synthetic biology, on a scale of 1 (not knowledgeable) to 5 (highly knowledgeable). By gathering user feedback from as many possible numeric combinations as possible, we wanted to test whether or not ezSTEP was considered usable regardless of the level of expertise in the two disciplines.

Additionally, we decided to split our user testing procedure into two stages (described in the sub-sections that follow). We have opted for this approach in order to be able to incorporate early user feedback as part of our platform's design. Since our interface design is user-oriented, incorporating early feedback could help improve the overall user experience for future users, making it more enjoyable.

5.4.1.1 Stage 1 User Testing

In Stage 1 user testing, we asked a small group of people to use the ezSTEP platform and monitored the way in which they interacted with it. They were provided with a short set of instructions and asked to fill in a survey on our app's usability. However, after they had answered all the questions in the survey and submitted their answers, we asked them for any suggestions that they felt could have made their experience more enjoyable. We then made a note of their suggestions and incorporated them into our design.

5.4.1.2 Stage 2 User Testing

In Stage 2 testing, the platform was made available to a larger audience, where we asked participants to fill in the same survey as in Stage 1, only this time without providing participants with an instructions sheet and monitoring their interaction. This was done to measure the ease with which users can understand how to interact with

our app and get a more accurate insight into how enjoyable the user experience actually is (i.e., are users getting easily frustrated with the platform, in which case this may need further investigation, or is the overall process smooth and users can easily adapt to using the app).

5.4.2 Results

As part of our survey, we asked each respondent to answer a series of questions, where, for each question, they were given a statement for which they could select an option from a scale of 'Strongly agree' to 'Strongly Disagree'. The results based on the participants' answers to these questions were generally positive and encouraging (see Appendix C for a detailed breakdown). One major insight we obtained was that users tend to have a pleasant experience when interacting with our platform, and there seemed to be an overall agreement on the fact that the model outputs interface and the interactive graphs were a strong suit of our platform.

Additionally, dividing the testing procedure into two stages has led to us gaining very useful feedback from early users. One such insight was the fact that the guidelines we provided for the users were very text-heavy, and a suggestion for improvement was to include some images as visualisation prompts to make the body of text less intimidating to read through. Implementing this change resulted in more positive feedback on the usefulness of the guidelines from respondents in the second stage of user testing.

Besides the series of questions, we also asked each participant to report the top 3 words that best described their user experience and thoughts on the platform. We collated these results to create a word cloud, which we show in Figure 5.8.

Figure 5.8: Word cloud generated using the answers from all participants in the user study.

Chapter 6

Conclusions

This chapter provides a final discussion of the work presented throughout this dissertation, the conclusions drawn, and the potential for future work. We also include information on how to access the platform as well as all the additional resources we used for evaluation (e.g., examples of datasets used for measuring model performance).

6.1 Accessibility

ezSTEP can be accessed using two methods. First, the most user-friendly way is to use the hosted application, which is currently hosted on Heroku: https://ezstep-f617792399bb.herokuapp.com. The landing page launches the home dashboard of the tool, which begins with some guidelines for the users, and further down the same page the user can find the input boxes to supply their own data. However, in the future, the ezSTEP platform will be hosted on a virtual machine, and will be accessible through the following URL: http://calbuco.inf.ed.ac.uk/ezSTEP

Alternatively, users can find instructions on how to set up the platform and run it locally using the GitHub repository here: https://github.com/AndreasHiropedi/ezSTEP. The instructions in the repository include how to install all the dependencies needed in order to successfully run the app, as well as the command for launching the dashboard locally.

We also include the training and testing datasets used for measuring the performance of our models, as well as the Python code used for obtaining those datasets from the larger original data. This information can also be found in the GitHub repository of our platform (https://github.com/AndreasHiropedi/ezSTEP), and we chose to include it for reproducibility of our results.

6.2 Limitations

Given the scope of the project, as well as the restricted timeline, several aspects of the project were limited. For instance, the current model selection for out platform focuses on models widely used in the literature. However, despite these models showing good

performance, this limited model selection inhibits the scope of the project's objective of providing a set of models that can achieve accurate results regardless of the input data.

Another limitation would be the lack of dependency control. For instance, our platform is highly reliant on the models provided by the scikit-learn Python package (Pedregosa et al., 2011); however, if the underlying implementation details of this package were to change significantly, our platform would not be robust to these changes.

Yet another limitation could be the user interface design/ implementation. Our current platform was designed to be used on laptop and PC-sized screens (or larger). However, the layout of the page is significantly affected if users try to access our platform on their mobile phones or tablets.

Additionally, summative evaluation aimed at the target audience was unable to be fully conducted: only a small proportion of the people involved in the user study were part of the actual target audience, namely biology researchers. That being said, the results of the user study were positive, meaning that the platform has achieved its objective of being usable regardless of user expertise. Nonetheless, these results should be considered only as assumptions that represent an estimated assessment of responses from the target audience.

6.3 Future Work

The strong performance of the models on several datasets, as well as the positive feedback following the usability testing, suggest that the platform has the potential for adoption amongst biology researchers. However, there is still significant room for improvement.

One aspect that can be explored in more detail is the integration of additional machine learning models with our platform. Since our research noted that tree-based model architectures, such as that of Random Forest, resulted in the highest overall performance across all datasets, one obvious model addition would be that of Extreme Gradient Boosting, also known as XGBoost (T. Chen & Guestrin, 2016).

Another aspect that could be further explored is that of reducing our dependency on the scikit-learn package (Pedregosa et al., 2011). Although more challenging to implement, a solution to this could be developing our own, in-house version of the models already used. Since we already have an API set up for each of the models (see section 4.2 in Chapter 4), we would only need to modify existing functions to achieve the desired functionality and remove the scikit-learn dependency.

Yet another possible extension of our platform could lie in expanding its scope: currently, ezSTEP only handles the problem of regression. However, by incorporating a mechanism that allows our web-app to also handle the problem of classification, we can provide a more powerful tool that combines the previous work done by platforms such as iLearnPlus (Z. Chen et al., 2021) and BioAutoMATED (Valeri et al., 2023) with that of our own, whilst retaining our original, user-friendly interface.

6.4 Concluding Remarks

Throughout this dissertation, we discussed the creation, development and evaluation of a new platform for measuring protein expression. ezSTEP has been designed on the premise of having a simple, usable platform that can successfully and accurately tackle the task of regression for different parts of DNA using small datasets and shallow machine learning models. Our user evaluation revealed the potential for adoption of our platform amongst the scientific community of biology researchers. The contributions of this project are as follows:

1. Build on the work done by Nikolados et al. (2022) to provide a new online platform for automatically training and testing some of the models covered in their original paper.
2. Conduct a literature review to identify existing platforms that handle sequence-to-expression classification tasks and account for limitations in their design.
3. Create a conceptual design that is not subject to the limitations previously identified and implement that design to build the first freely accessible platform that addresses the task of sequence-to-expression regression.
4. Evaluate the performance of this platform across different datasets, as well as assess its usability through a user study with human participants. This is a novel approach that has not been used by existing platforms to assess their usability.
5. Acknowledge the potential for further development of the platform and further studies to improve the tool and extend the evaluation.

Bibliography

Alipanahi, B., Delong, A., Weirauch, M. T., & Frey, B. J. (2015). Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, *33*(8), 831–838.

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., ... Farhan, L. (2021). Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, *8*, 1–74.

Angenent-Mari, N. M., Garruss, A. S., Soenksen, L. R., Church, G., & Collins, J. J. (2020). A deep learning approach to programmable rna switches. *Nature communications*, *11*(1), 5057.

Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L., & Ridella, S. (2012). The'k'in k-fold cross validation. In *Esann* (Vol. 102, pp. 441–446).

Avsec, Ž., Kreuzhuber, R., Israeli, J., Xu, N., Cheng, J., Shrikumar, A., ... others (2019). The kipoi repository accelerates community exchange and reuse of predictive models for genomics. *Nature biotechnology*, *37*(6), 592–600.

Biehl, M. (2015). *Api architecture* (Vol. 2). API-University Press.

Bisong, E., & Bisong, E. (2019). An overview of google cloud platform services. *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, 7–10.

Cambray, G., Guimaraes, J. C., & Arkin, A. P. (2018). Evaluation of 244,000 synthetic sequences reveals design principles to optimize translation in escherichia coli. *Nature biotechnology*, *36*(10), 1005–1015.

Carlson, J. (2013). *Redis in action*. Simon and Schuster.

Chai, T., & Draxler, R. R. (2014). Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. *Geoscientific model development*, *7*(3), 1247–1250.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794).

Chen, Z., Zhao, P., Li, C., Li, F., Xiang, D., Chen, Y.-Z., ... Song, J. (2021, 02). iLearnPlus: a comprehensive and automated machine-learning platform for nucleic acid and protein sequence analysis, prediction and visualization. *Nucleic Acids Research*, *49*(10), e60-e60.

Copeland, M., Soh, J., Puca, A., Manning, M., Gollob, D., Copeland, M., ... Gollob, D. (2015). Microsoft azure and cloud computing. *Microsoft Azure: Planning, Deploying, and Managing Your Data center in the Cloud*, 3–26.

Cuperus, J. T., Groves, B., Kuchina, A., Rosenberg, A. B., Jojic, N., Fields, S., & Seelig, G. (2017). Deep learning of the regulatory grammar of yeast 5' untranslated regions from 500,000 random sequences. *Genome research*, *27*(12), 2015–2024.

Fei, N., Gao, Y., Lu, Z., & Xiang, T. (2021). Z-score normalization, hubness, and few-shot learning. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 142–151).

He, X., Zhao, K., & Chu, X. (2021). Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, *212*, 106622.

Höllerer, S., Papaxanthos, L., Gumpinger, A. C., Fischer, K., Beisel, C., Borgwardt, K., ... Jeschek, M. (2020). Large-scale dna-based phenotypic recording and deep learning enable highly accurate sequence-function mapping. *Nature communications*, *11*(1), 3551.

Hopkin, K., Johnson, A., Morgan, D., Raff, M., Roberts, K., & Walter, P. (2018). *Essential cell biology: Fifth international student edition*. W.W. Norton.

Jaganathan, K., Panagiotopoulou, S. K., McRae, J. F., Darbandi, S. F., Knowles, D., Li, Y. I., ... others (2019). Predicting splicing from primary sequence with deep learning. *Cell*, *176*(3), 535–548.

KIreeva, M., Trang, C., Matevosyan, G., Turek-Herman, J., Chasov, V., Lubkowska, L., & Kashlev, M. (2018, June). RNA–DNA and DNA–DNA base-pairing at the upstream edge of the transcription bubble regulate translocation of RNA polymerase and transcription rate. *Nucleic Acids Research*, *46*(11), 5764–5775. doi: 10.1093/nar/gky393

Kirk, J. M., Kim, S. O., Inoue, K., Smola, M. J., Lee, D. M., Schertzer, M. D., ... others (2018). Functional classification of long non-coding rnas by k-mer content. *Nature genetics*, *50*(10), 1474–1482.

Kotopka, B. J., & Smolke, C. D. (2020a). Model-driven generation of artificial yeast promoters. *Nature communications*, *11*(1), 2113.

Kotopka, B. J., & Smolke, C. D. (2020b, December). Model-driven generation of artificial yeast promoters. *Nature Communications*, *11*(1), 2113. doi: 10.1038/s41467-020-15977-4

LaCroix, J. (2018). *Mastering ubuntu server: Master the art of deploying, configuring, managing, and troubleshooting ubuntu server 18.04*. Packt Publishing Ltd.

Meier, B. (2019). *Python gui programming cookbook: Develop functional and responsive user interfaces with tkinter and pyqt5*. Packt Publishing Ltd.

Middleton, N., & Schneeman, R. (2013). *Heroku: up and running: effortless application deployment and scaling*. " O'Reilly Media, Inc.".

Miles, J. (2005). R-squared, adjusted r-squared. *Encyclopedia of statistics in behavioral science*.

Moore, A. W. (2001). Cross-validation for detecting and preventing overfitting. *School of Computer Science Carneigie Mellon University*, *133*.

Niedenthal, R., Riles, L., Johnston, M., & Hegemann, J. (1996). Green fluorescent protein as a marker for gene expression and subcellular localization in budding yeast. *Yeast*, *12*(8), 773–786.

Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. In *Proceedings of the sigchi conference on human factors in computing systems* (pp.

152–158).

Nikolados, E.-M., & Oyarzún, D. A. (2023). Deep learning for optimization of protein expression. *Current Opinion in Biotechnology*, *81*, 102941. doi: 10.1016/j.copbio.2023.102941

Nikolados, E.-M., Wongprommoon, A., Aodha, O. M., Cambray, G., & Oyarzún, D. A. (2022, December). Accuracy and data efficiency in deep learning models of protein expression. *Nat. Commun.*, *13*(1), 7755.

Okada, S., Ohzeki, M., & Taguchi, S. (2019). Efficient partition of integer optimization problems with one-hot encoding. *Scientific reports*, *9*(1), 13036.

Olson, R. S., & Moore, J. H. (2016). Tpot: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning* (pp. 66–74).

Patro, S., & Sahu, K. K. (2015). Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Penzar, D., Nogina, D., Meshcheryakov, G., Lando, A., Rafi, A. M., de Boer, C., ... Kulakovskiy, I. V. (2022, December). LegNet: Resetting the bar in deep learning for accurate prediction of promoter activity and variant effects from massive parallel reporter assays. , 2022.12.22.521582. doi: 10.1101/2022.12.22.521582

Pisner, D. A., & Schnyer, D. M. (2020). Support vector machine. In *Machine learning* (pp. 101–121). Elsevier.

Richards, T. (2021). *Getting started with streamlit for data science: Create and deploy streamlit web applications from scratch in python*. Packt Publishing Ltd.

Rodriguez-Galiano, V., Sanchez-Castillo, M., Chica-Olmo, M., & Chica-Rivas, M. (2015). Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines. *Ore Geology Reviews*, *71*, 804–818.

Roell, M.-S., & Zurbriggen, M. D. (2020, February). The impact of synthetic biology for future agriculture and nutrition. *Current Opinion in Biotechnology*, *61*, 102–109. doi: 10.1016/j.copbio.2019.10.004

Saeys, Y., Inza, I., & Larranaga, P. (2007). A review of feature selection techniques in bioinformatics. *bioinformatics*, *23*(19), 2507–2517.

Sample, P. J., Wang, B., Reid, D. W., Presnyak, V., McFadyen, I. J., Morris, D. R., & Seelig, G. (2019). Human 5' utr design and variant effect prediction from a massively parallel translation assay. *Nature biotechnology*, *37*(7), 803–809.

Shah, P., Ding, Y., Niemczyk, M., Kudla, G., & Plotkin, J. B. (2013, June). Rate-Limiting Steps in Yeast Protein Translation. *Cell*, *153*(7), 1589–1601. doi: 10.1016/j.cell.2013.05.049

Sinha, V. K., Vaarties, K., De Buck, S. S., Fenu, L. A., Nijsen, M., Gilissen, R. A. H. J., ... Smit, J. W. (2008). Towards a better prediction of peak concentration, volume of distribution and half-life after oral drug administration in man, using allometry. *PubMed*, *47*(1), 35-45. Retrieved from `https://pubmed.ncbi.nlm.nih.gov/18076217` doi: 10.2165/00003088-200847010-00004

Spadiut, O., Capone, S., Krainer, F., Glieder, A., & Herwig, C. (2014, January). Mi-

crobials for the production of monoclonal antibodies and antibody fragments. *Trends in Biotechnology*, *32*(1), 54–60. doi: 10.1016/j.tibtech.2013.10.002

Timpl, L., Entezari, R., Sedghi, H., Neyshabur, B., & Saukh, O. (2022). Understanding the effect of sparsity on neural networks robustness. *arXiv preprint arXiv:2206.10915*.

Vaishnav, E. D., de Boer, C. G., Molinet, J., Yassour, M., Fan, L., Adiconis, X., ... Regev, A. (2022). The evolution, evolvability and engineering of gene regulatory dna. *Nature*, *603*(7901), 455–463.

Valeri, J. A., Soenksen, L. R., Collins, K. M., Ramesh, P., Cai, G., Powers, R., ... others (2023). BioAutoMATED: An end-to-end automated machine learning tool for explanation and design of biological sequences. *Cell Systems*, *14*(6), 525–542.

Wang, X., Sun, K., Batcheller, A., & Jajodia, S. (2020). An empirical study of secret security patch in open source software. *Adaptive Autonomous Secure Cyber Systems*, 269–289.

Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, *30*(1), 79–82.

Wittig, A., & Wittig, M. (2023). *Amazon web services in action: An in-depth guide to aws*. Simon and Schuster.

Wittmann, B. J., Yue, Y., & Arnold, F. H. (2021). Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell systems*, *12*(11), 1026–1045.

Wong, K.-C., Li, Y., & Zhang, Z. (2016). Unsupervised learning in genome informatics. *Unsupervised learning algorithms*, 405–448.

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 8697–8710).

Appendix A

Nielsen's Usability Heuristics

A.1 Heuristic 1: Visibility of system status

The design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time.

A.2 Heuristic 2: Match between system and the real world

The design should speak the users' language. Use words, phrases, and concepts familiar to the user, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.

A.3 Heuristic 3: User Control and Freedom

Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action without having to go through an extended process.

A.4 Heuristic 4: Consistency and Standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform and industry conventions.

A.5 Heuristic 5: Error prevention

Good error messages are important, but the best designs carefully prevent problems from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

A.6 Heuristic 6: Recognition rather than recall

Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design (e.g. field labels or menu items) should be visible or easily retrievable when needed.

A.7 Heuristic 7: Flexibility and efficiency of use

Shortcuts — hidden from novice users — may speed up the interaction for the expert user so that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

A.8 Heuristic 8: Aesthetic and minimalist design

Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility

A.9 Heuristic 9: Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no error codes), precisely indicate the problem, and constructively suggest a solution.

A.10 Heuristic 10: Help and documentation

It's best if the system doesn't need any additional explanation. However, it may be necessary to provide documentation to help users understand how to complete their tasks.

Appendix B

Participants' Information Sheet and Consent Form

B.1 Information Sheet

This study was certified according to the Informatics Research Ethics Process, reference number 867658. Please take time to read the following information carefully. The title of the project involved is "ezSTEP: an online platform for training and testing machine learning models for protein expression", and the lead researchers in this study are Dr. Diego Oyarzún, Yuxin Shen, and Andreas Hiropedi.

What is the purpose of the study?

Assess the usability of the web-app platform.

Why have I been asked to take part?

The research target group is a mixture of students with high expertise, limited expertise, and no expertise, in order to get a good sense of how easy to use the platform is based on knowledge in the field. Our aim is to ensure the platform is user-friendly and easy to use regardless of the level of knowledge.

Do I have to take part?

No – participation in this study is entirely up to you. You can withdraw from the study at any time, without giving a reason. Your rights will not be affected. If you wish to withdraw, contact the PI, Andreas Hiropedi. We will stop using your data in any publications or presentations submitted after you have withdrawn consent. However, we will keep copies of your original consent, and of your withdrawal request.

What will happen if I decide to take part?

If you decide to take part, you will be asked to access a website hosting our online platform and play around on the platform to try all the available features. Once you have experimented for a good period of time (e.g., say 10-15 minutes), you will be asked to fill out a Microsoft form, where you will need to indicate your level of expertise, as well as share your personal thoughts on your experience using the platform. You will

not be recorded during the study, and you can do the study completely unsupervised and in your own time.

Are there any risks associated with taking part?

There are no significant risks associated with participation.

Are there any benefits associated with taking part?

No

What will happen to the results of this study?

The results of this study may be summarised in published articles, reports and presentations. Quotes or key findings will be anonymized: We will remove any information that could, in our assessment, allow anyone to identify you. Once we have compiled the data and added it to the final report, all files storing information will be deleted immediately.

Data protection and confidentiality.

Your data will be processed in accordance with Data Protection Law. All information collected about you will be kept strictly confidential. We will compile the data into groups based on level of expertise, and so no names or any information will be included besides simply the level of expertise you indicate in the form. Your data will only be viewed by the researcher/research team, which strictly includes only my supervisor, my supporting PhD student, and myself. All electronic data will be stored on a password-protected encrypted computer, on the School of Informatics' secure file servers, or on the University's secure encrypted cloud storage services (DataShare, ownCloud, or Sharepoint) and all paper records will be stored in a locked filing cabinet in the PI's office. Your consent information will be kept separate from your responses in order to minimise risk.

What are my data protection rights?

You have the right to access information held about you. Your right of access can be exercised in accordance Data Protection Law. You also have other rights including rights of correction, erasure and objection. For more details, including the right to lodge a complaint with the Information Commissioner's Office, please visit www.ico.org.uk. Questions, comments and requests about your personal data can also be sent to the University Data Protection Officer at dpo@ed.ac.uk. For general information about how we use your data, go to: https://data-protection.ed.ac.uk/privacy-notice-research

Who can I contact?

If you have any further questions about the study, please contact the lead researcher, Andreas Hiropedi, s2015345@ed.ac.uk. If you wish to make a complaint about the study, please contact inf-ethics@inf.ed.ac.uk. When you contact us, please provide the study title and detail the nature of your complaint.

B.2 Declaration of Consent

By agreeing to take part in this study, you agree that you have read and understood the information presented above, you understand that your participation is voluntary, and are happy to have your answers considered, processed and publicly summarised (as part of a greater assessment). No personal information that can help identify you, as the respondent, will be collected or made publicly available (we are only interested in your level of knowledge, a very objective measure that cannot be used to easily identify someone, and so your data will be anonymised); however, your answers will be used to create summary statistics about the overall trend of all responses collected.

Do you consent to taking part in this study?

1. Yes
2. No

B.3 Platform Usability Survey

The link to the survey used for our platform usability study can be accessed using the following link: https://forms.office.com/e/aNijwbb3wH

Appendix C

Results of the User Study

In this appendix, we provide a detailed breakdown of the results of our platform's usability study. We have split this appendix into eight sections, where each section corresponds to a specific question on the survey we sent out to our participants (see the end of Appendix B).

C.1 Level of Expertise

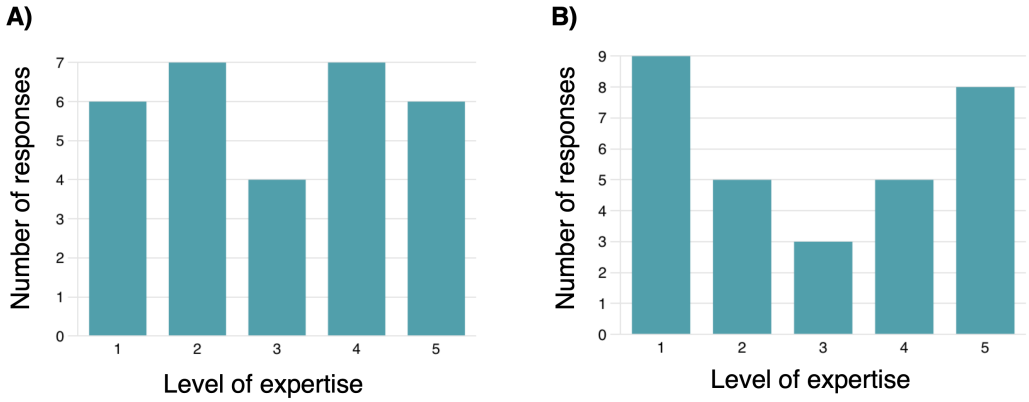


Figure C.1: A) The aggregated responses across all user study participants indicating their expertise in programming. **B)** The aggregated responses across all user study participants indicating their expertise in synthetic biology.

After agreeing to take part in our survey, the first question we asked our participants was to indicate their level of expertise in programming and synthetic biology on a scale of 1 (not knowledgeable) to 5 (highly knowledgeable). We show the aggregated answers to this question in Figure C.1.

As can be seen from the figure, the answer distribution for expertise in programming (Figure C.1 A) shows that we have tested our platform on an audience with a highly varied level of knowledge in programming. These results could mimic the landscape

of the scientific community of biology researchers, as some researchers are more experienced with using tools such as machine learning models for sequence-to-expression prediction tasks.

On the other hand, the answer distribution for expertise in synthetic biology (Figure C.1 B) is more skewed towards the extreme ends (1 and 5). This could suggest that further testing may be necessary to ensure our positive results are in fact indicative of our platform's potential for adoption amongst the scientific community of biology researchers: we would ideally want a distribution that is left-skewed, indicating a high level of expertise in synthetic biology.

C.2 Usefulness of the User Guidelines

Figure C.2: The aggregated responses across all user study participants for the statement 'I found the user guidelines to be helpful for providing information on how to use the app'.

Once the users input their levels of expertise, they are then prompted to experiment with the platform and then share their opinions about a series of statements. The first such statement is:

I found the user guidelines to be helpful for providing information on how to use the app.

The results for this question are presented in Figure C.2. As we can see, users generally found that the guidelines provided useful insight into the features offered by ezSTEP. It is also worth mentioning that, despite the positive results, one early piece of feedback that we received suggested we include visualisations in the guidelines, as well as reduce their size. The feedback we originally received was that the guidelines were 'too large' and 'very intimidating to read', hence the high number of 'Neutral' responses. We therefore incorporated this feedback into our refined version, which received much better feedback (most of the answers with 'Strongly agree' were obtained following this change).

C.3 User Interface Design

The next statement that we asked users to evaluate is related to the actual design of our platform. The statement we gave users was:

I found the user interface of the app to be highly interactive and easy to use.

Figure C.3: The aggregated responses across all user study participants for the statement 'I found the user interface of the app to be highly interactive and easy to use'.

The collected responses to this statement are shown in Figure C.3. These results are encouraging, with over 80% of the users selecting either 'Agree' or 'Strongly Agree', and we can safely attribute these results to our user-oriented design when building and developing the platform.

C.4 Model Inputs Page Design

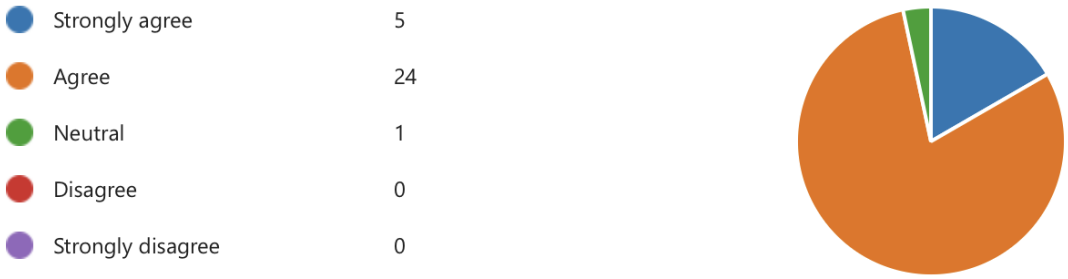


Figure C.4: The aggregated responses across all user study participants for the statement 'I found the individual model input pages highly interactive and easy to use'.

The statement that we will focus on in this section has to do with evaluating the design of our platform's individual input pages. The statement we gave users was:

I found the individual model input pages highly interactive and easy to use.

The results for this question are presented in Figure C.4. Although the vast majority of our users selected the 'Agree' option, we changed a feature on this page at the very beginning of our study after our first response was 'Neutral'. This feature was related

to how users were being updated on the model creation process (informed on whether the model was still training or the model had been created successfully). The early feedback allowed us to implement a quick change that ensured a much smoother user experience, as suggested by the other positive responses.

C.5 Input Parameters Selection

In addition to evaluating the design of our individual input pages, we also asked for users' opinions on the available selection of input parameters. To do so, we used the following statement:

I found the breadth of choice in terms of model parameters selection to be nice and provide great opportunities for experimentation.

The collected responses to this statement are shown in Figure C.5. The results show that we provided enough variation for users to be able to customize their sequence-to-expression models, but not to the extent that users feel overwhelmed by the vast amount of choices.

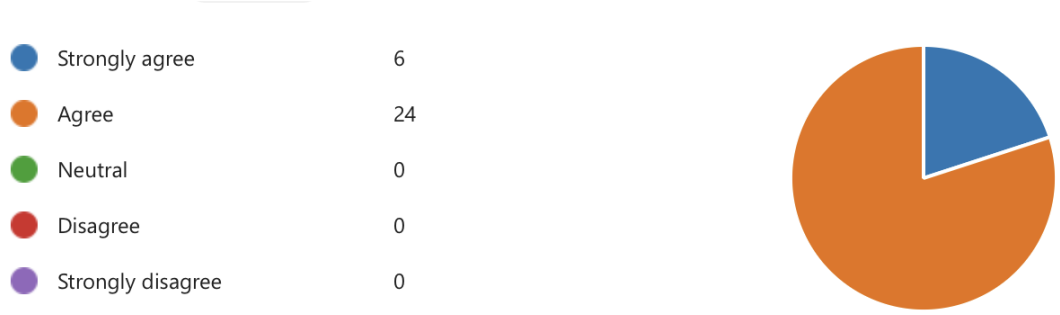


Figure C.5: The aggregated responses across all user study participants for the statement 'I found the breadth of choice in terms of model parameters selection to be nice and provide great opportunities for experimentation'.

C.6 Model Outputs Page Design

Besides the individual input pages, we also wanted to evaluate the corresponding individual output pages. We used the statement below as a prompt in our survey:

I found the individual model outputs pages to provide plenty of useful information for analysis and comparison with other models.

Our findings for this question are presented in Figure C.6. The results here are overall positive, with 30% of the users selecting the 'Strongly agree' option and 67% the 'Agree' option. We have also discussed the design of this page with researchers in the university's Biomolecular Control Group, and the feedback on the page's layout suggested our design helps provide a clear structure for the results being displayed.

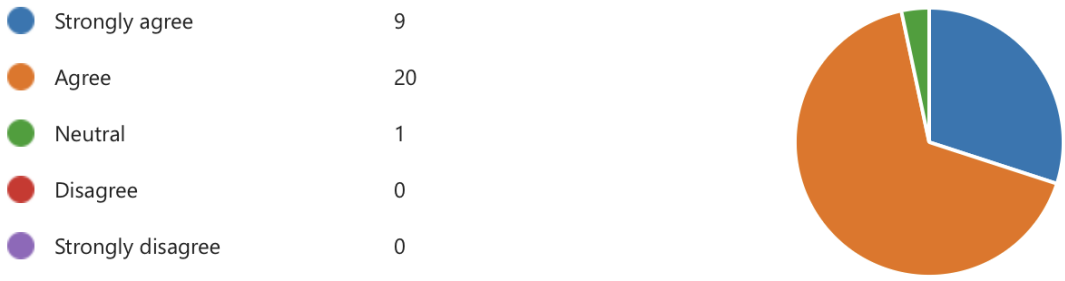


Figure C.6: The aggregated responses across all user study participants for the statement 'I found the individual model outputs pages to provide plenty of useful information for analysis and comparison with other models'.

C.7 Interactive Features of the Output Graphs

One aspect of our app we felt was particularly important was the high degree of interactivity of our graphs: we wanted users to freely experiment with the results we provide, and tweak them in any way necessary for their analysis. To assess this, we provided users with the following statement:

I found all graphs in the app to be user-friendly and highly interactive, enabling users to experiment with all the available features (such as enabling/ disabling different parts of the graph, saving them on their local devices, etc.).

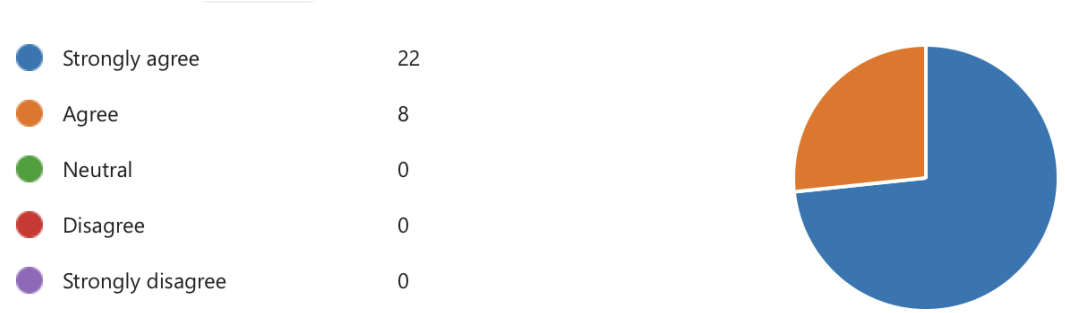


Figure C.7: The aggregated responses across all user study participants for the statement 'I found all graphs in the app to be user friendly and highly interactive, enabling users to experiment with all the available features (such as enabling/ disabling different parts of the graph, saving them on their local devices, etc.)'.

The collected responses to this statement are shown in Figure C.7. The results suggest that this is the strongest aspect of our app, with nearly 70% of responses being 'Strongly agree'. This suggests that the highly interactive graphs provide a very enjoyable experience for the users, who are able to freely explore the results in more detail and customize the graphs to their liking.

C.8 Overall User Experience

Lastly, after analysing each individual component of our app, we wanted to get our users' final thoughts on their overall experience. This would not only enable us to evaluate whether the app is usable or not, but we could also notice patterns (based on previous answers) about which individual features may have more impact on user experience over others. The statement given to our participants was:

I found the overall process of using the app to be pleasant and enjoyable.

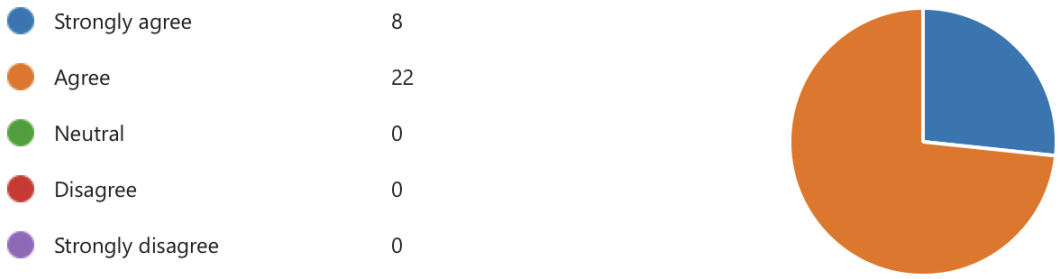


Figure C.8: The aggregated responses across all user study participants for the statement 'I found the overall process of using the app to be pleasant and enjoyable'.

The results for this question are presented in Figure C.8. We note that all our answers are either 'Agree' or 'Strongly agree', suggesting that the participants in our study viewed our platform as relatively user-friendly and easy to use. We observed that the features with the most importance in this decision, especially when differentiating between 'Agree' and 'Strongly Agree', were the helpfulness of the user guidelines, the breadth of choice in input parameters, and the high degree of interactivity of our graphs, all of which interestingly seemed to impact user experience more than the design of our GUI.

Appendix D

Pre-print manuscript

In this appendix, we provide the full contents of the pre-print mentioned in the abstract, formatted exactly as it would be for submission to the ACS Journal for Synthetic Biology.

ezSTEP: a web-based tool for training and testing sequence-to-expression models

Andreas Hiropedi,[1] Yuxin Shen,[2] and Diego A. Oyarzún[1, 2, 3]
[1)] *School of Informatics, University of Edinburgh, UK*
[2)] *School of Biological Sciences, University of Edinburgh, UK*
[3)] *Corresponding author: d.oyarzun@ed.ac.uk*

(Dated: 1 April 2024)

Abstract: We present ezSTEP (**S**equence-**T**o-**E**xpression **P**redictor), a software tool that can be used for protein expression prediction from DNA sequences using shallow machine learning models. Our user-friendly platform allows users to create, customise, train, and evaluate sequence-to-expression models using their own data, by simply uploading it onto the web-app. It can also be used as a tool for data exploration and model optimisation, as well as generating protein expression predictions for unlabeled sequences. This paper presents ezSTEP and its features through specific use cases, showcasing its practical applications.

Keywords: DNA sequence, machine learning, training, customising, querying, protein expression, data exploration

I. INTRODUCTION

Microbial engineering involves the manipulation of microbes in order to develop novel functions, with wide applications in the pharmaceutical, food and energy sectors[13,15]. Predicting protein expression in the cells from DNA sequences is critical for strains optimization in heterologous protein production[9]. However, this has proven to be notoriously challenging, as protein expression involves the complex transcriptional and translational process in biosystems[7,14].

Current advances in high-throughput sequencing and screening techniques have facilitated the acquisition of large sequence-to-expression datasets. Machine learning models can be built on these datasets to deliver protein expression prediction of acceptable accuracy for practical applications. Previous research has demonstrated that convolutional neural networks (CNN) can achieve an R-squared prediction accuracy of approximately 0.9 for ribosome binding sequences (RBS)[5]. Machine learning models have also be trained to predict protein expression from promoter performance[8,12]. Deep learning models applied to 5'-UTR sequences can facilitate the in silico evolution of DNA[3]. Nonetheless, such models often suffer from certain limitations when it comes to their usability and interpretability, posing great hurdles and barriers for biology researchers.

Considering these limitations, efforts have been made in order to ease the adoption of machine learning tools to analyze biological datasets. A proposed solution that has proven to be quite promising is automated machine learning (AutoML), which automates the design and deployment of ML pipelines with minimal user intervention[4]. Several tools that adopt this AutoML architecture have already been made publicly available. For instance, BioAutoMATED provides users with an end-to-end pipeline for biological sequence design[17]. Another example would be iLearnPlus, a web platform for DNA, RNA and protein sequence classification, which wraps together different feature extraction methods and classifiers[2]. Vaishnav et al. also created a web platform that enables users to query a pre-trained transformer model which was trained on 20 million promoter sequences[16].

In this paper, we introduce a new platform, namely ezSTEP (Sequence-To-Expression Predictor). Our aim is to provide a user-friendly platform for biologists on sequence-to-expression regression tasks. ezSTEP also adopts the AutoML architecture, allowing non-programmers to easily create, train, and evaluate their own sequence-to-expression models in a simple and efficient way.

II. EZSTEP

ezSTEP is a tool that can either be accessed through a web browser or downloaded for private installation (see the Accessibility section for details). The platform allows users to build and optimise machine learning models on their own datasets as well as query these models.

Figure 1A describes the pipeline of the ezSTEP platform. The DNA sequences and corresponding expression data initially go through a preprocessing stage, where features from the data are extracted. These features are used by the model in order to learn how to accurately estimate the expression levels. After preprocessing, there are two optional steps, namely feature extraction and hyperparameter optimisation. These steps are only executed if the user enables them as part of their model input selection. The last step before model training is model selection, where the user chooses the type of machine learning algorithm to be used. Once the model has been successfully trained, its performance is then measured on the provided test set as part of the model evaluation stage. The last step in the pipeline, namely model querying, is optional, and will only be executed if the user supplied a query dataset.

ezSTEP takes training and testing datasets as required

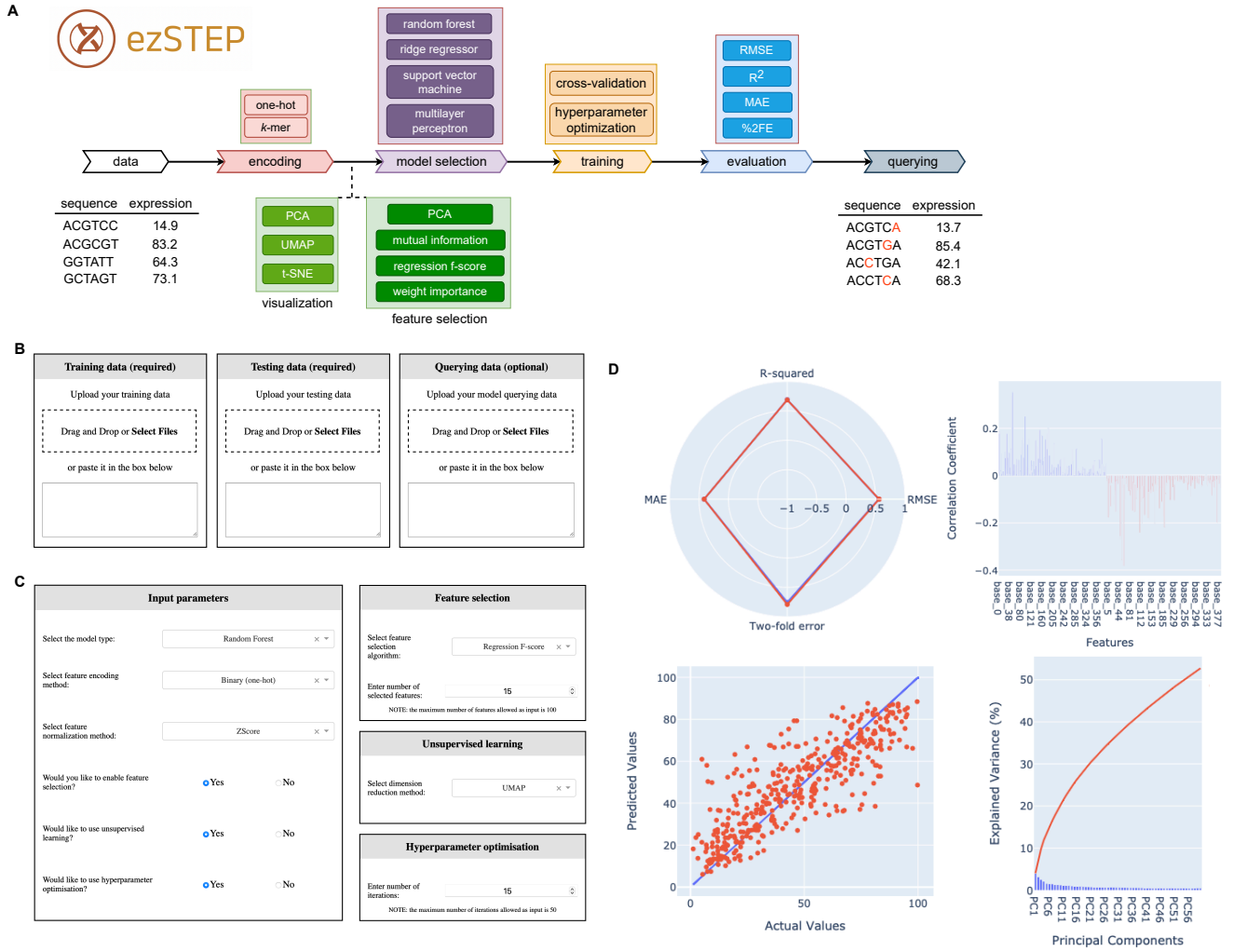


FIG. 1. **A)** The working pipeline of ezSTEP platform. **B)** The upload boxes where users can upload their own training, testing and querying data. **C)** The model input page, where the users can customize their sequence-to-expression models. **D)** The model output page, where the users are provided with interactive graphs for evaluating the performance of the models as well as providing insights into the model's architecture.

inputs, and the querying sequences as an optional input. The datasets can be passed in either as a file, with the accepted format being CSV, or entered manually into a text box (see Figure 1B). Once the user provides this data, it is then validated to ensure that it is in the correct format. This includes simple checks such as ensuring correct formatting, as well as checking for irregularities in the data itself, such as DNA sequences not being of equal length or illegal characters being included in the DNA sequences.

The sequence-to-expression data can be obtained from various experimental conditions in different labs, and can also be on different parts of the DNA sequence, like ribosome binding sites and promoters. This could cause the data to vary significantly across different labs, resulting in the inability to extend existing models to predict sequences using unseen data, and hence limiting researchers to using their own in-house models. In order to ensure that ezSTEP is not subject to similar pitfalls, we have tested our platform using three different datasets, each designed for different parts of the DNA sequence.

Once the user has inputted their data, they can then proceed to creating their customized sequence-to-expression models. Users can create as many models as they wish, and can also delete models they created. Figure 1C shows the user interface for customizing sequence-to-expression models, and we provide a short summary of the available choices for each potential input parameter below:

1. **Model type**: Random Forest, Multi-layer Perceptron, Support Vector Machine, Ridge Regressor. These models are built using the scikit-learn package in Python[11].

2. **Feature encoding**: one-hot encoding or k-mer (k = 2-5)
3. **Feature normalisation**: MinMax or Z-score
4. **Feature selection (optional)**: requires the following two inputs:
   (a) **feature number**: a number between 1 and 100 indicating the number of features to be selected for training
   (b) **feature selection algorithm**: a choice between PCA, Weight importance, Regression f-score and Mutual Information
5. **Unsupervised learning (optional)**: a choice between PCA, UMAP or t-SNE
6. **Hyper-parameter optimisation (optional)**: uses Bayesian optimisation, requires one input:
   (a) **iteration number**: a number between 1 and 50 indicating the number of iterations that the optimisation is run for

After providing inputs for the components outlined above, the user can submit these inputs in order to create, train and test their model by pressing the blue 'Submit model selection' button (see Figure 1C). Once pressed, the platform first checks that all the needed inputs have been provided (no missing or invalid values), and also ensures that training and testing datasets have been provided and validated successfully. If these checks pass, the model is successfully trained and tested in real time.

Once the training process has finished, the user will have access to several plots that provide more insight into the model's performance and its underlying architecture (see Figure 1D). All these plots are generated using the plotly package in Python[6], and provide several user-interactive features, such as the ability to enable/ disable visualisations for certain components of the graph, zooming in and out, and saving the graph as a PNG file to the user's local device. We provide a short summary of the available graphs on the ezSTEP platform below:

1. A plot showing the training statistics versus testing statistics (for ease of visualisation, since numbers in a table may be harder to interpret).
2. An actual values versus model predictions plot.
3. Plots showing the correlation between different input features (extracted from the DNA sequences) and the target variable (the protein expression) for the training and test data.
4. If the user enables feature selection, a plot showing the explained variance of the selected features.
5. If the user enables unsupervised learning, a PCA, t-SNE or UMAP plot (depending on the user's choice of unsupervised learning algorithm) analysing the provided data.

In the remainder of this paper, we describe how the main functionalities of ezSTEP outlined in this section can be used in practice, by considering a series of use cases.

A. Use case 1: Model training, optimisation and querying on the Cambray dataset

The first dataset used to evaluate the functionality and performance of our platform was a dataset created by Cambray et al. (2018), with synthetic DNA sequences for the translation optimisation in *Escherichia coli (E. coli)*. The GFP reporter fluorescence was evaluated for 244,000 sequences on the 5' coding region of DNA, spreading across 56 different mutational series of the DNA sequence space[1].

Given the level of diversity of this dataset, we selected one of the 56 mutational series of the DNA sequence space in order to assess the performance of our models. We randomly sampled roughly 4000 sequences from this mutational series, and then applied a 90/10 split on the sampled sequences to create our training and test datasets.

After creating these datasets and curating them to ensure compatibility with our platform, we then created several models with different input configurations (using our platform's user interface shown in Figure 1C) and measured their performance. In these experiments we varied model type, feature encoding and normalization method, as well as enabling additional features such as feature selection and hyper-parameter optimisation. To assess a model's performance, we compute the following four metrics: root mean squared error (RMSE), R-squared, mean absolute error (MAE), and percentage (%) within 2-fold error[5].

Results from the ezSTEP is shown in Figure 2. As we can see, the model's performance on this dataset is quite good, with R-squared values of as high as 0.77 on the test data for the Random Forest model (see Figure 2A). Additionally, we see that this performance is further enhanced when hyper-parameter optimisation is enabled: originally, the Support Vector Machine model achieves a R-squared value of 0.71, but this value is increased to 0.81 when the model is further optimised. We further compared these results with a research paper based on the Cambray dataset[10], and found that the R-squared results from ezSTEP's baseline models (without optimisation) align with the findings in this paper, and even surpass the paper's findings when optimisation is enabled.

B. Use case 2: Dataset Exploration

Despite the promising results mentioned in the previous section, as stated at the start of this paper, one of ezSTEP's main objectives is to provide users with models that can generalise well to a wide variety of data.

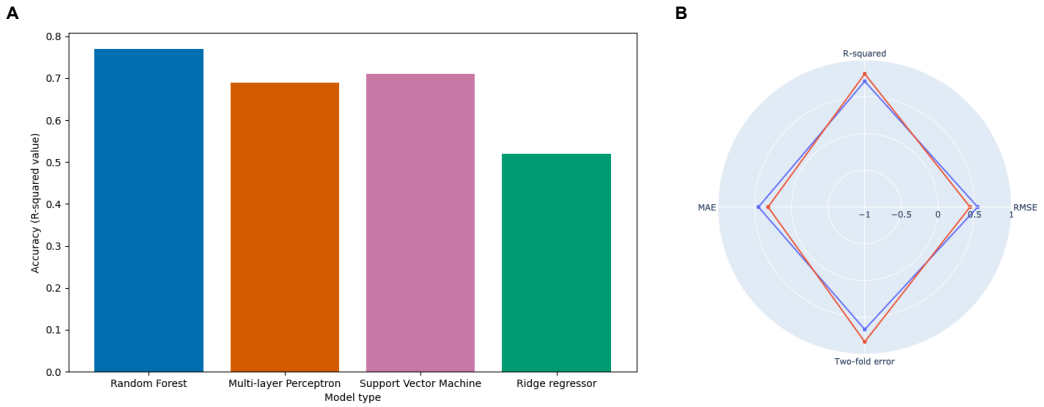


FIG. 2. **A)** Recorded R-squared values for four different models on the test set. The data used for obtaining these values came from the dataset created by Cambray et al. (2018). **B)** Performance of the support vector machine model on the Cambray dataset with (red) and without (blue) hyper-parameter optimisation.

Since we allow users to upload their own data, we needed to ensure that our models can achieve similarly promising results on different DNA strains. We have therefore tested our platform using two additional datasets, and compared ezSTEP's performance with the corresponding research papers.

The first additional dataset consisted of natural yeast promoter sequences, and originated from the Vaishnav et al. (2022) paper. This data was originally used as a validation set for the pre-trained model in the original paper[16]. Nikolados et al. (2022) measured the performance of shallow machine learning models on this dataset. The promoter sequences are 80 nucleotides long, and were designed to optimize translation in *Saccharomyces cerevisiae*. Similar to the Cambray dataset, we randomly sampled around 4000 sequences, and then performed a 90/10 split to create our training and test datasets.

Our second additional dataset originated from the Höllerer et al. (2020) paper. This dataset contains around 300,000 bacterial RBS sequences[5], which are much shorter than in the previous two datasets, with a length of only 17 nucleotides. We once again randomly sampled around 4000 sequences given the large scale of the original data, and then performed a 90/10 split on the sampled sequences to create our training and test datasets.

After using the generated training and test datasets to assess our platform's performance on these two additional DNA strains, we observed that our models' performance, although different from that on the Cambray dataset, aligned with the results in their corresponding research papers[5,10]. Nonetheless, since the performance of the models inevitably varies when using different datasets, we concluded that further insights into the structure of the data could help clarify the reasons behind the measured performance.

These insights can be seen by enabling unsupervised learning and selecting one of the available options (PCA, UMAP, or t-SNE). When enabling this option, an additional plot will be displayed on the model output page. In the case of the UMAP plot, the graph is colour encoded based on the protein expression level, giving an insight of the correlation between output and input structure. To allow the users to explore such insights in depth, we made these graphs highly interactive and gave users the ability to change the values of key parameters in the unsupervised learning algorithm itself. By changing these parameters, users can automatically regenerate the plot in real time.

We showcase these plots, as well as illustrate their importance, in Figure 3. We can clearly see from Figure 3A that, when the parameters are adjusted to appropriate values, they reveal important insights in the data: from the plot on the left, we would incorrectly assume that the data is quite randomly spread out, but a closer look shows that the data in fact comprises of two unique clusters, as show in the plot on the right.

With the appropriate UMAP parameters respectively for each dataset, we visualised the RBS, coding and promoter sequences datasets Figure 3B-D). The UMAP of coding sequence shows that there is no significant correlation between the two clusters and the protein expression level (Figure 3B). The UMAP of RBS sequence shows that the points in the big cluster are more likely to have lower protein expression, while points spreading around tend to have higher protein expression Figure 3C). The UMAP of natural promoter sequence shows that the sequences are highly clustered, and the sequences in one cluster have similar protein expression level (Figure 3D). These three examples showcase how visualisation of the input data enhances the understanding of the dataset, and has potential in troubleshooting the model.

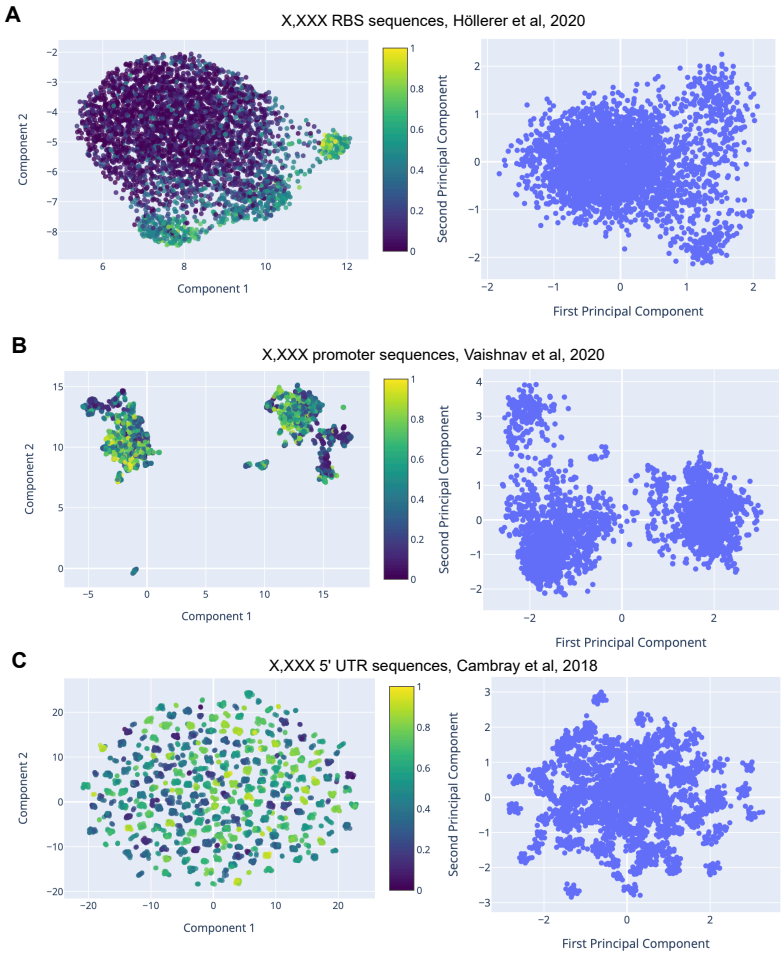


FIG. 3. **A)** UMAP transition when changing the algorithm's input parameters. **B)** The UMAP (left) and PCA (right) plot for only the training data obtained from the coding sequences in the dataset in the Cambray et al. (2018) paper. **C)** The UMAP (left) and PCA (right) plot for only the training data obtained from the ribosome binding sites (RBS) sequences in the dataset in the Höllerer et al. (2020) paper. **D)** The UMAP (left) and PCA (right) plot for only the training data obtained from the promoter sequences in the dataset in the Vaishnav et al. (2022) paper.

III. ACCESSIBILITY

ezSTEP can be accessed using two methods. First, the most user-friendly way is to use the hosted application: `https://ezstep-f617792399bb.herokuapp.com`. The landing page launches the home dashboard of the tool, which begins with some guidelines for the users, and further down the same page the user can find the input boxes to supply their own data.

Alternatively, users can find instructions on how to set up the platform and run it locally using the repository here: `https://github.com/AndreasHiropedi/ezSTEP`. The instructions in the repository include how to install all the dependencies needed in order to successfully run the app, as well as the command for launching the dashboard locally.

ACKNOWLEDGEMENTS

YS was supported by the UKRI Biotechnology and Biological Sciences Research Council (BBSRC) grant number BB/T00875X/1. DAO was supported by the United Kingdom Research and Innovation (grant EP/S02431X/1), UKRI Centre for Doctoral Training in Biomedical AI.

REFERENCES

[1] Guillaume Cambray, Joao C Guimaraes, and Adam Paul Arkin. Evaluation of 244,000 synthetic sequences reveals design principles to optimize translation in escherichia coli. *Nature biotechnology*, 36(10):1005–1015, 2018.

[2] Zhen Chen, Pei Zhao, Chen Li, Fuyi Li, Dongxu Xiang, Yong-Zi Chen, Tatsuya Akutsu, Roger J Daly, Geoffrey I Webb, Quanzhi

Zhao, Lukasz Kurgan, and Jiangning Song. iLearnPlus: a comprehensive and automated machine-learning platform for nucleic acid and protein sequence analysis, prediction and visualization. *Nucleic Acids Research*, 49(10):e60–e60, 02 2021.

[3] Josh T. Cuperus, Benjamin Groves, Anna Kuchina, Alexander B. Rosenberg, Nebojsa Jojic, Stanley Fields, and Georg Seelig. Deep learning of the regulatory grammar of yeast 5′ untranslated regions from 500,000 random sequences. *Genome Research*, 27(12):2015–2024, January 2017.

[4] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

[5] Simon Höllerer, Laetitia Papaxanthos, Anja Cathrin Gumpinger, Katrin Fischer, Christian Beisel, Karsten Borgwardt, Yaakov Benenson, and Markus Jeschek. Large-scale dna-based phenotypic recording and deep learning enable highly accurate sequence-function mapping. *Nature communications*, 11(1):3551, 2020.

[6] Plotly Technologies Inc. Collaborative data science. `https://plot.ly`, 2015.

[7] Maria KIreeva, Cyndi Trang, Gayane Matevosyan, Joshua Turek-Herman, Vitaly Chasov, Lucyna Lubkowska, and Mikhail Kashlev. RNA–DNA and DNA–DNA base-pairing at the upstream edge of the transcription bubble regulate translocation of RNA polymerase and transcription rate. *Nucleic Acids Research*, 46(11):5764–5775, June 2018.

[8] Benjamin J. Kotopka and Christina D. Smolke. Model-driven generation of artificial yeast promoters. *Nature Communications*, 11(1):2113, December 2020.

[9] Evangelos-Marios Nikolados and Diego A Oyarzún. Deep learning for optimization of protein expression. *Current Opinion in Biotechnology*, 81:102941, 2023.

[10] Evangelos-Marios Nikolados, Arin Wongprommoon, Oisin Mac Aodha, Guillaume Cambray, and Diego A. Oyarzún. Accuracy and data efficiency in deep learning models of protein expression. *Nature Communications*, 13(1):7755, December 2022.

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

[12] Dmitry Penzar, Daria Nogina, Georgy Meshcheryakov, Andrey Lando, Abdul Muntakim Rafi, Carl de Boer, Arsenii Zinkevich, and Ivan V. Kulakovskiy. LegNet: Resetting the bar in deep learning for accurate prediction of promoter activity and variant effects from massive parallel reporter assays. page 2022.12.22.521582, December 2022.

[13] Marc-Sven Roell and Matias D Zurbriggen. The impact of synthetic biology for future agriculture and nutrition. *Current Opinion in Biotechnology*, 61:102–109, February 2020.

[14] Premal Shah, Yang Ding, Malwina Niemczyk, Grzegorz Kudla, and Joshua B. Plotkin. Rate-Limiting Steps in Yeast Protein Translation. *Cell*, 153(7):1589–1601, June 2013.

[15] Oliver Spadiut, Simona Capone, Florian Krainer, Anton Glieder, and Christoph Herwig. Microbials for the production of monoclonal antibodies and antibody fragments. *Trends in Biotechnology*, 32(1):54–60, January 2014.

[16] Eeshit Dhaval Vaishnav, Carl G. de Boer, Jennifer Molinet, Moran Yassour, Lin Fan, Xian Adiconis, Dawn A. Thompson, Joshua Z. Levin, Francisco A. Cubillos, and Aviv Regev. The evolution, evolvability and engineering of gene regulatory DNA. *Nature*, 603(7901):455–463, March 2022.

[17] Jacqueline A. Valeri, Luis R. Soenksen, Katherine M. Collins, Pradeep Ramesh, George Cai, Rani Powers, Nicolaas M. Angenent-Mari, Diogo M. Camacho, Felix Wong, Timothy K. Lu, and James J. Collins. BioAutoMATED: An end-to-end automated machine learning tool for explanation and design of biological sequences. *Cell Systems*, 14(6):525–542.e9, June 2023.