Norwegian University
of Life Sciences

Master's Thesis 2023 30 ECTS
Faculty of Science and Technology

Evaluation of Machine Learning Methods to Decode Transcriptional Regulation

Harini Jeyakumar
Data Science

Acknowledgement

Firstly, I want to thank all my supervisors for their guidance and help during the work of this thesis. I want to thank my main supervisor Dean, Torgeir Rhoden Hvidsten for introducing me to this topic. I also want to thank my co-supervisors Prof. Simen Rød Sandve and Researcher Lars Grønvold for sharing their knowledge in biology and for giving me advice. In addition, I would like to thank my co-supervisor Prof. Kristian Hovde Liland for technical help whenever I needed it.

Furthermore, I would like to thank my family and close friends for their support, especially my roommates for interesting ideas throughout this semester.

Harini Jeyakumar, May 2023

Abstract

With large biological measurements made possible by the development of high-throughput technology, it allows for the study of genomic data. In transcriptional regulation the cell controls the translation of DNA to RNA, and thereby controls which genes to express. Exploring the regulatory mechanisms underlying the genes that are controlled in a cell is part of epigenetic profiling. Finding accessible DNA regions that control transcriptional regulation can be done using this method. It appears that machine learning models have not previously been tested on ATAC-STARR-seq data with varying fragment length from the salmon genome. Using the results from an ATAC-STARR-seq experiment on a salmon genome, we want to explore the extent to which it is possible to predict from sequence fragments.

In this thesis, we attempt to predict to what degree sequence fragments can drive transcription using the ATAC-STARR-seq results of over five million sequence fragments. Fragments within the top 10% basemean were selected for the model's training, validation, and testing in order to improve the performance of the machine learning models. Techniques such as feature engineering and feature importance were crucial for training the model and extracting relevant features for motif search. We evaluated three classical machine learning methods for regression analysis predicting log2FoldChange values. The ensemble methods XGBoost Regressor and Random Forest Regressor, and Linear Support Vector Regression were applied as machine learning algorithms. XGBoost Regressor and Random Forest Regressor are both powerful algorithms known to have been used successfully on sequential data. Linear Support Vector Regression's ability to handle a large number of samples compared to Support Vector Regression, was one of the reasons this algorithm was chosen.

Furthermore, XGBoost Regressor and Random Forest Regressor performed remarkably similar with potential for improvement. The Linear Support Vector Regression model had more trouble capturing the complexity of the data. Despite the results, it was possible to extract sequence features from the trained machine learning models and associate them with known transcription factors. This study's findings indicate that there is still a need for improvements in the performance of the models. While time limit and time-consuming algorithms have been a challenge, possibilities for tuning with not yet tested parameters and other methods to improve the model remains for further research.

Contents

Preface **i**

Abstract **iii**

Table of Contents **iv**

List of Figures **vi**

1 Introduction **1**

1.1 Motivation . . . 1
1.2 Objectives . . . 2
1.3 Structure . . . 2

2 Theory **3**

2.1 DNA . . . 3
2.1.1 Transcription and Translation . . . 4
2.1.2 Gene Regulation . . . 4
2.2 ATAC-STARR-seq . . . 6
2.3 Machine Learning . . . 7
2.3.1 Supervised ML problems . . . 8
2.3.2 Decision Tree Regression . . . 8
2.3.3 Random Forest Regressor . . . 9
2.3.4 XGBoost regressor . . . 10
2.3.5 Linear Support Vector Regression . . . 12
2.3.6 Metrics . . . 12
2.3.7 Related works . . . 13

3 Method **14**

3.1 Dataset . . . 14
3.2 Preprocessing . . . 14
3.2.1 Data Cleaning . . . 15
3.2.2 Splitting the dataset . . . 15
3.2.3 Filtering Basemean . . . 15
3.2.4 Feature Engineering . . . 16
3.2.5 Train, Validation and Test . . . 16
3.3 XGBoost Regressor Model . . . 17
3.3.1 Hyperparametertuning . . . 17
3.3.2 Feature importance . . . 17
3.3.3 Motif search . . . 18

3.4 Random Forest Regressor Model 18
3.4.1 Hyperparametertuning 18
3.5 Linear SVR 19
3.5.1 Hyperparametertuning 19

4 Results **20**
4.1 XGBoost Regressor Default VS Tuned 20
4.2 Random Forest Regressor: Default VS Tuned 24
4.3 Linear SVR: Default VS Tuned 26
4.4 Motif Search Discovery 27

5 Discussion **30**
5.1 Data Quality 30
5.2 Model Performance 30
5.3 Future Work 32

6 Conclusion **33**

A Table of Python libraries **37**

List of Figures

2.1 Illustration of the structure of a DNA double helix. Figure used under a CC0 license [3]. 4
2.2 Illustration of transcriptional regulation in eukaryotes. Figure used under a CC BY-SA 4.0 license[11]. 6
2.3 Illustration of the 5-fold cross-validation. Inspired by Koehrsen [17] 8
2.4 Illustration of decision tree for regression. The features are k-mer count vectors and the predicted values are log2FoldChange values 9
2.5 Illustration of the making of Random Forest Regressor. Each decision tree's prediction is averaged, and a final result is given. The figure is inspired by Chaya [20] 10
2.6 Illustration of the gradient boosting method. Adding trees into the model, helps to lower model error. Figure is inspired by Saha [24] 12

3.1 Preprocessing steps for every model 15
3.2 16

4.1 Default XGB Regressor 20
4.2 Default XGB Regressor 21
4.3 XGB Regressor Learning Curve 22
4.4 The tuned XGB Regressor scatterplot 23
4.5 Default Random Forest Regressor scatterplot 24
4.6 Tuned Random Forest Regressor model 25
4.7 Default Linear SVR scatterplot 26
4.8 The tuned Linear SVR scatterplot 27
4.9 Transcription factors HNF4A and FOXA1 28
4.10 Transcription factors RXRalpha and FOXA2 29

List of Tables

3.1 Dimension (shape) of training, validation and test set 17
3.2 Hypermarameters tuned in RandomizedSearchCV with repspective values and the chosen parameters for XGBoost Regressor tuned model 18
3.3 Hyperparameters tuned in RandomizedSearchCV with respective values and the chosen parameters for Random Forest Regressor tuned model 18
3.4 Hyperparameters tuned in GridSearchCV with respective values and the chosen parameters for Linear SVR tuned model 19

4.1 Metrics performance for default XGBoost Regressor Model 21
4.2 Metrics performance for tuned XGBoost Regressor Model 23
4.3 Metrics performance for default Random Forest Model 24
4.4 Metrics performance for tuned Random Forest Model 25
4.5 Metrics performance for default Linear SVR 26
4.6 Metrics performance for tuned Linear SVR 27

A.1 Python packages used in this study 37

Chapter 1

Introduction

The whole collection of genetic information present in each cell of an organism is known as the genome. It contains every piece of knowledge required for an organism to grow and operate, and thereby gives the instructions for who we are. In transcriptional regulation, the cell controls the translation of DNA to RNA. This process, also referred to as gene regulation, includes choosing which genes to express. A gene is a fragment of the DNA sequence that can be translated into a protein, and thus has a biological function in the cell. It is not possible for all genes to be expressed constantly, under all circumstances, in all cell types of tissues, and in all cells. Controlling when and how much genes are expressed is crucial in biological processes. Various diseases such as cancer can be brought on when transcription is improperly regulated. Thus, it is important to understand how transcriptional regulation functions in a genome.

With the development of high-throughput technologies, large biological measurements allow for the analysis of genomic data. DNA Sequencing, transcription profiling and epigenetic profiling are a few examples of such technologies. Epigenetic profiling explores the regulatory mechanisms behind genes that are regulated in a cell [1]. This can be used to find accessible DNA regions that may drive transcriptional regulation. The DNA regions are enriched of promoters and enhancers which are regulatory regions crucial for regulating the expression of genes. A variety of methods, including machine learning, can be used to analyse the genome. By using machine learning techniques it may be possible to predict to what degree sequence fragments can drive transcription. Identifying fragments by analysing the features and patterns of the regions that drive transcription can be challenging. The prediction of regulatory activity in eukaryotic species has been proposed using different machine learning methods.

1.1 Motivation

Transcriptional regulation plays an important role in how genetic information moves across cells. It is of interest to know how cis-regulatory elements such as enhancers are connected and coded in the genome.

Sandve et al. (unpublished results) has conducted an ATAC-STARR-seq experiment on the salmon genome, in liver cells, where millions of fragments were captured. The STARR-seq data, which contains information on regulatory activity in fragments, depends on the sequence of the fragments. Based on the sequences, it could be possible to use machine learning to predict regulatory activity.

Numerous sequence fragments of various lengths from the salmon genome through ATAC-STARR-Seq have not previously been tested on machine learning models. By extracting biological knowledge from datasets using machine learning, we may discover what kind of function the various fragments have, as well as the circumstances under which they are upregulated and downregulated. It would also allow predicting the effect of sequence variants. We may thus evaluate how it impacts salmon health and growth, making this relevant for among other things, environmental research. As there will be more similar experiments in the future, it is useful to gain a better knowledge of how a genome can be utilized.

Machine learning models can be used to predict the ability of regulatory regions to drive transcription, according to the study by Wang et al.[2]. There have been published several papers that use machine learning techniques in addition to this study, which includes using Support Vector Machines and Naïve Bayes.

1.2 Objectives

The aim of this master thesis is to thoroughly evaluate three classical machine learning approaches for predicting to what degree sequence fragments from ATAC-STARR-seq can drive transcription. In that case, the sequence features that drive transcription could be extracted and be associated with known transcription factor binding sites.

1.3 Structure

This thesis begins with introducing the theoretical background needed to understand the method and results. We cover the theory of transcriptional regulation, the ATAC-STARR-seq method and machine learning methods used in this thesis. Chapter 3 presents the dataset, data preprocessing and how the models are implemented. Chapter 4 presents the results, and we discuss them in Chapter 5. The source code can be found in the GitHub repository https://github.com/harinjey/thesis-data-science, with separate Python files for each default and tuned model. Table A.1 in Appendix A lists each package used in Python, along with its versions.

Chapter 2

Theory

In this chapter we will cover the theory on biology, the ATAC-STARR-Seq method used to generete the dataset and the machine learning models implemented in this thesis.

2.1 DNA

The genetic information for all organisms is carried in the deoxyribonucleic acid (DNA) on a chromosome of a cell nucleus. A section of DNA called a gene can be found on a chromosome, and the instructions to produce a protein are encoded in most genes.

DNA is made up of nucleotides, and has a double helix-like structure with two complementary strands. The key components of the DNA molecule, known as nucleotides, consists of a phosphate group, deoxyribose sugar, and a nitrogen base. Adenine, guanine, cytosine, and thymine are the four types of nitrogen bases. They are ordered in a specific sequence. Adenine and thymine bond together, while cytosine and guanine similarly form bonds. The two complementary strands are held together by hydrogen bonds formed between the bases. Each complementary DNA strand has a specific direction, from the 5'-to-3' end, and the two strands lie in the opposite direction from each other.

Adenine and thymine have two hydrogen bonds, while Cytosine and Guanine have three (Fig. 2.1). Because of this, these bases create hydrogen bonds with one another rather than randomly binding to each other. The bases are usually abbreviated to A, T, C and G.

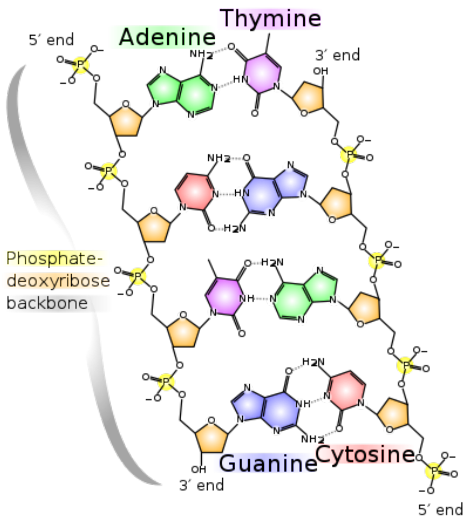

Figure 2.1: Illustration of the structure of a DNA double helix. Figure used under a CC0 license [3].

2.1.1 Transcription and Translation

In order for cells to be able to perform important tasks such as forming important structures in the cell and control immune response, it needs to produce proteins based on the genetic information in DNA [4]. This process is called protein synthesis. Transcription and translation are two processes involved in protein synthesis.

During transcription, DNA serves as a template for the production of ribonucleic acid (RNA) by copying into messenger RNA (mRNA) [5]. Most of the bases in RNA are the same as in DNA, except for thymine, which is replaced by the base uracil. mRNA carries the instructions for producing proteins. The backbone of RNA molecules is made up of alternating phosphate groups and the sugar ribose. All living cells normally have single-stranded RNA. Before the translation process, Non-coding segments called introns are removed from the mRNA. The RNA strand then only consists of exons, which are the coding segments in the genes. The mRNA is then given a cap at each end to prevent damage before the translation process begins [5].

Translation is a process that involves translating RNA into aminoacids with the help of ribosomes[5]. A ribosome is an assisting structure that connects amino acids in the order shown by codons from the mRNA. An adaptor molecule, called transfer RNA (tRNA) is needed for mRNA to produce proteins. Three nucleotides make up an mRNA codon, but a single codon encodes all of the information for an amino acid. When mRNA passes through the ribosome, each codon interacts with the corresponding anticodon of a tRNA molecule. The tRNA carries an amino acid which will be connected to a growing polypeptide [5].

2.1.2 Gene Regulation

Any processes that regulate how genes are expressed are referred to as gene regulation. Since Atlantic salmon dataset is used in this study, eukaryotic gene regulation will be the main focus. The term eukaryote refers to a cell that contains a nucleus.

Gene regulation makes it possible for every cell in an organism to have various sets of genes that can be ”turned” on (expressed) and off, which gives various patterns of gene expression. As a result, each cell gets different protein sets, which is necessary for the cell to do particular tasks [6].

Which genes a cell expresses can be influenced by a variety of factors. The information from inside a cell and the environment can affect the pattern of gene expression in a cell, therefore it is possible that two different cells of the same type can express different genes. In order to change the expression of a gene, cells contain molecular pathways that translate information into gene expression. Molecular pathways can act on receptors and change the gene expression in the cell.[6].

In eukaryotes, the chromatin structure, transcription and translation are all examples of processes at which the expression of genes can be regulated. Chromatin is a combination of DNA and proteins that makes up the chromosomes found in eukaryotic cells [7]. A closed chromatin is more densely packed, and an open chromatin is less densely packed, i.e. more accessible for transcription. It is possible to regulate the structure of chromatin. For instance, an open chromatin can increase the gene availability for transcription [6] .

Transcriptional regulation

As previously mentioned, proteins are synthesised through the processess of transcription and translation. By regulating which genes to transcribe into mRNA, the cell can control how much protein is made from a gene, and thereby develop and function properly [8]. This process is known as transcriptional regulation.

The enzyme RNA polymerase, which creates a new RNA molecule using a DNA template, must bind to a section of the gene known as a promoter with the assistance of transcription factors in order for a gene to be transcribed. Transcription factors are proteins essential for transcriptional regulation and can turn on genes in a cell. The regulatory elements required to regulate how the gene's transcription is started are found in the promoter, which is located upstream of a gene. Promoters can work with other DNA regions to ensure that the gene is properly transcribed [9]. An example of such a region is enhancers, also a regulatory region, which interact with promoters to facilitate transcription [10]. Another regulatory region are silencers which in contrast to enhancers, reduce the transcription rate and downregulates the expression of a gene.

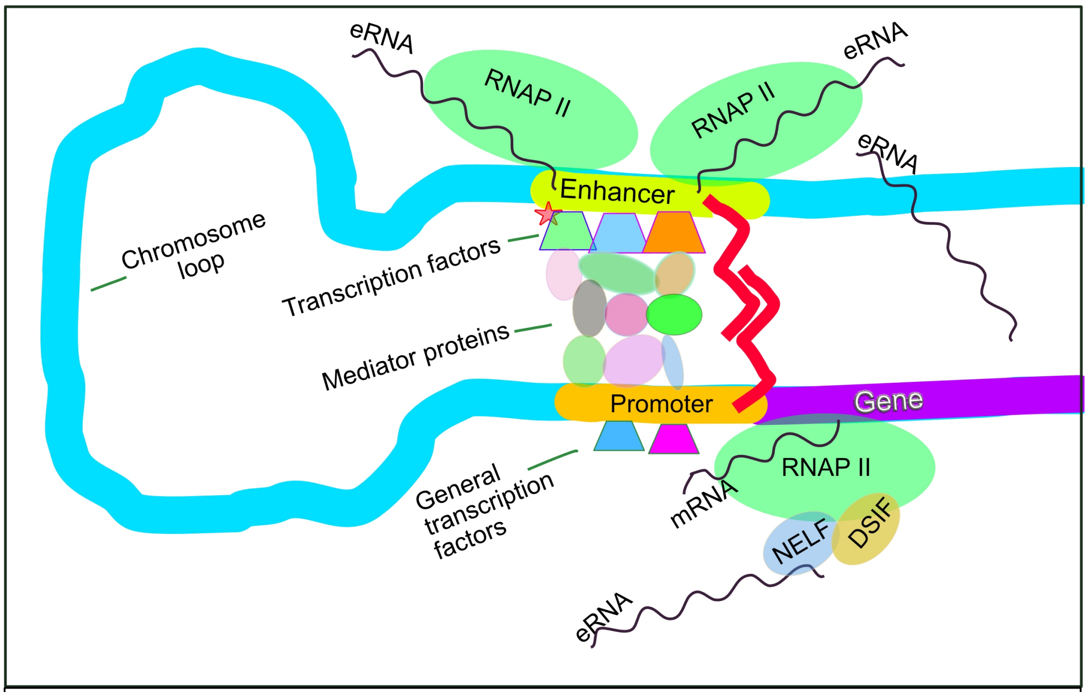

Figure 2.2: Illustration of transcriptional regulation in eukaryotes. Figure used under a CC BY-SA 4.0 license[11].

2.2 ATAC-STARR-seq

Within epigenetic profiling techniques, there is a tool called ATAC-STARR-seq that can be used to identify sequences in the genome that have regulatory activity, such as enhancer activity. ATAC-STARR-seq is a massively parallel reporter assay (MPRA) that combines the ATAC-seq and STARR-seq methods. By using the assay transposase-accessible chromatin with high throughput sequencing (ATAC-seq) method, fragments enriched in regulatory regions can be extracted from open chromatin and sequenced. Tn5 transposase, a DNA-cutting enzyme, is used to extract the fragments from chromatin. ATAC fragments are used as input sequences to the STARR assay. Other options than ATAC fragments, would be to use designed sequences which are expensive, or random sequences for the entire genome, which would include all the uninteresting regions [2].

The ATAC fragments can then be inserted into the 3' end of a reporter gene plasmid. This is done throughout the method called Self-Transcribing Active Regulatory Region sequencing, referred to as STARR-seq. The enhancer activity of the ATAC fragments is measured by STARR-seq. When the plasmid library is generated, it is transfected into a cell culture, and the reporter gene is then transcribed into mRNA by the cell's transcription machinery. Enhancer activity in the sequences of the fragments can result in higher expression and the amount of mRNA will increase. By using targeted RNA-seq to sequence the mRNA, the regulatory activity can be measured. During the library preparation process, Tn5 tags are added to the plasmid containing the reporter gene. To amplify DNA or RNA sequences, a method known as Polymerase Chain Reaction (PCR) is utilized. Since the cell produces its own mRNA, it can cause the measurement of enhancer activity to be inaccurate. PCR will target the Tn5 tags and make sure that only mRNA from the plasmid with the reporter gene are sequenced. The input plasmid library can contain a higher or lower copy number of certain fragments. Since this can have an impact on how the results are interpreted, the library is sequenced as a control with DNA sequencing.

Following that, while processing sequencing data using bioinformatics, the number of reads or observations for each fragment in DNA and RNA is counted. Using the software package BWA, the sequenced reads are mapped to the genome of the Atlantic salmon. The number of unique fragments is counted using a custom Python script, where unique fragments are read-pairs that align to the same position. Mismatches are ignored. The ratio of each unique fragment's RNA and DNA copies is used to determine the regulatory activity. To calculate this, the count data is used as input in DESeq2. DESeq2 is a tool for analyzing differential gene expression in RNA-seq data between two or more samples, as well as hypothesis testing [12]. The input plasmid library DNA-seq is used as the control sample and the output RNA-seq is used as the tested sample. Five replicates are used to create the DNA-seq and RNA-seq. The results from the DESeq2 computation is the log2FoldChange, FDR adjusted p-value and basemean. The generated data and the sequence fragments can then be analysed by machine learning methods to identify regulatory elements that drive transcription.

2.3 Machine Learning

Computers are able to acquire knowledge without being specifically programmed thanks to an area of artificial intelligence known as machine learning. It enables a machine to learn from previously given data, while artificial intelligence enables a machine to replicate human intelligence to solve problems [13]. Machine learning makes use of algorithms that have been trained to classify and generate predictions to uncover insightful patterns in data. Speech recognition, chatbots, image recognition and healthcare advancement are a few applications of machine learning in everyday life [14].

Machine learning algorithms fall into three categories: supervised learning, unsupervised learning and reinforcement learning. Supervised learning techniques which will be the focus in this thesis, employ the model to make predictions on new data after learning it from labelled training data. The labelled training data consists of the input data/features and corresponding target values/labels. The goal is to train the machine learning model such that the algorithm can make calculations and learn the patterns between the input data and its corresponding value. The trained model can then be used for prediction on unseen data. By training the model, the accuracy of the model can be improved over time. For instance, if a model is trained with images of several alphabets, it will eventually learn to recognize each alphabet by itself. Unsupervised learning identifies patterns in unlabeled data. This approach is ideal for applications such as image and pattern recognition because it can identify similarities and differences without needing a labelled dataset [14]. Dimensionality reduction can be achieved through this learning method. Compressing features onto a lower dimensional subspace is possible with dimensionality reduction techniques. One can benefit from less storage space and faster computation time by reducing the dimensionality of the feature space. The performance of a model may be improved by removing noisy features. In reinforcement machine learning, an agent is trained through trial and error [15]. By using the response from the behaviour of the agent in an environment with tasks, the agent learns from its mistakes. The response can be positive or negative, corresponding to reward and punishment. The goal is to reward wanted behaviors, thereby maximizing the reward function [15].

A well-known problem in machine learning is model bias. It is a result of incorrect predictions made throughout the machine learning process. In other words, bias in a model indicates how well the model fits the training set of data. This concept may lead to a model with high bias that does not accurately fit the training data or a model with low bias that fits significantly better. The training data is the data used for training the model. High bias can be caused by underfitting, which occurs when a model is too simple to recognize patterns in the data and produce inaccurate predictions. In contrast, an overfitting model is too specific and performs well on the training data, making it difficult to generalize on test data. In order to create a general model for a machine learning task, a fair balance between under- and overfitting is needed [16].

Cross Validation

To detect overfitting, we go though the process called Cross Validation. After dividing the dataset into a training and test set, the method K-fold cross validation can be applied to the machine learning task. By dividing the training set into K-folds, which are smaller subsets of the training set, we run K separate experiments. One K-subset is chosen to serve as the test set in each experiment. The rest of the K-1 subsets (or folds) are combined to form the training set. After running the K separate experiments, the K different testing sets performances are averaged. A validation metrics for the model is then produced [17]. By performing K-fold cross-validation, we will get a more accurate measure of the performance of the model. One can compare the performances of the training and validation sets to detect overfitting. Figure 2.3 illustrates the process of a 5-fold cross-validation. Each experiment (iteration in the figure) uses a single K-subset as the validation set. In this thesis, 3-fold cross-validation is implemented in the RandomizedSearchCV and GridSearchCV for hyperparametertuning.

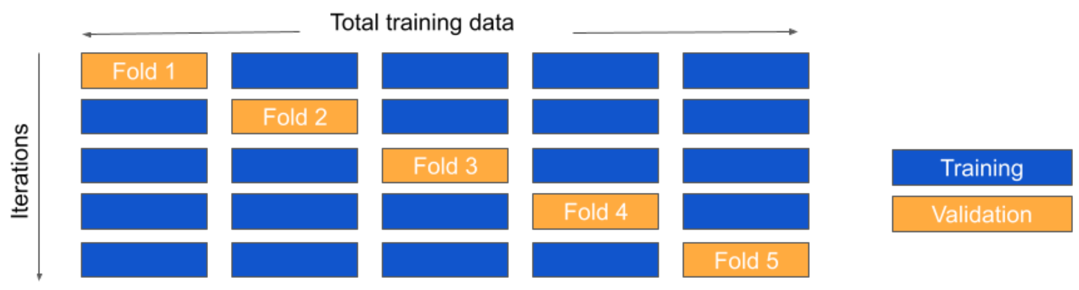

Figure 2.3: Illustration of the 5-fold cross-validation. Inspired by Koehrsen [17]

2.3.1 Supervised ML problems

Supervised machine learning can further be divided into two tasks: classification and regression. Classification is used to predict the class label of an object based on its features. Regression, which is the task in this thesis, predicts continuous values of new data by using the slope and intercept learnt from training data.

2.3.2 Decision Tree Regression

Decision Trees is a popular algorithm within supervised machine learning used in both classification and regression tasks. The model can be visualised as a tree with three different nodes. The root node represents the first decision, and is divided into two new nodes based on the value of the first feature, getting two new branches. This leads to the internal nodes, which make another decision based on the value of another feature. Decision trees select features and values recursively until they reach a leaf node. Every branch is selected based on whether the value of a feature meets a certain threshold. The predicted value is shown through the leaf nodes [18].

In decision tree regression, the predicted value is not a class as in classification, but a continuous value. Bias and overfitting are some of the challenges in this type of algorithm.

As mentioned, the algorithm works by dividing the data into two new nodes based on whether the value of a feature meets a certain threshold. While running through the tree, the algorithm fits a constant value to each set [18]. In the context of the thesis, the data observations, in this case DNA fragments, are divided based on k-mer count vectors, which are the features. Every new data observation will then get a predicted log2FoldChange value. Figure 2.4 illustrates a decision tree for regression. The k-mer feature of size 3 represents the first feature. This is divided depending on the log2FoldChange value of the feature. If the value is larger than 2.5, the data points will go to the branch on the right, k-mer feature of size 6. If not, the data points will go the the left branch, k-mer size of 4.

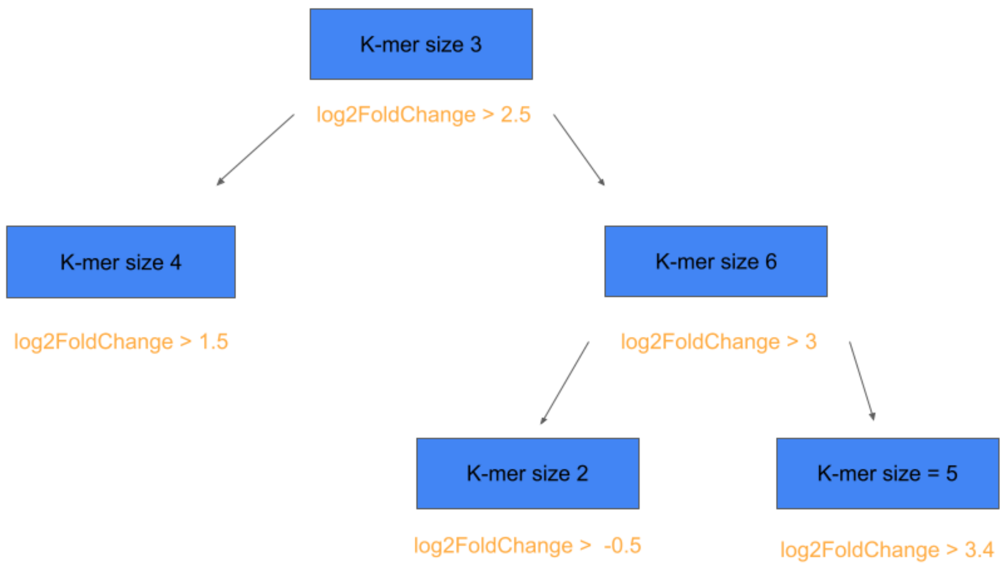

Figure 2.4: Illustration of decision tree for regression. The features are k-mer count vectors and the predicted values are log2FoldChange values

This subsection will provide an explanation of the first two machine learning regression algorithms that were selected for this thesis. Both of the algorithms are based on the decision tree regression algorithm.

2.3.3 Random Forest Regressor

If we look back at Decision Trees, the algorithm takes every feature into consideration when dividing the nodes. In contrast, the Random Forest algorithm only chooses a subset of the features. An ensemble of trees makes up the Random Forest model. In this method, the prediction is made based on every observation by each decision tree. At last, the prediction of the trees are merged together. Random Forest can be used to solve both classification and regression tasks. In classification, the predicted class label, which is the combination of each tree's classification, will be determined by what is called a majority vote. It assumes that each classifier/decision tree votes for a class, and the class with the most votes is chosen as the predicted class label. Unlike classification, the combined predictions (decision trees) are averaged when dealing with a regression task [19].

The random forest algorithm is based on the Bagging method. Bagging involves choosing different training data points from the training data with replacement. Every tree goes through this process, which will result in trees that have been trained on different data points. When a data observation gets replaced, it is possible to get different combinations of the various observations from the training set, thereby getting random samples. The out-of-bag sample, which is a third of the training data is used as test data [19].

By using the bagging method in Random Forest, it is possible to achieve low bias. Since the algorithm applies majority voting or averages predictions from each tree, the variance is lowered and overfitting is prevented. This algorithm is often chosen by Data Scientists as it gives high accuracy solving both supervised machine learning tasks. It will become clear in the following chapters that training this kind of model requires a lot of time. The reason is that each decision tree in the algorithm has to go through the process of predicting a classification label or a continuous value, making it a complex model [19].

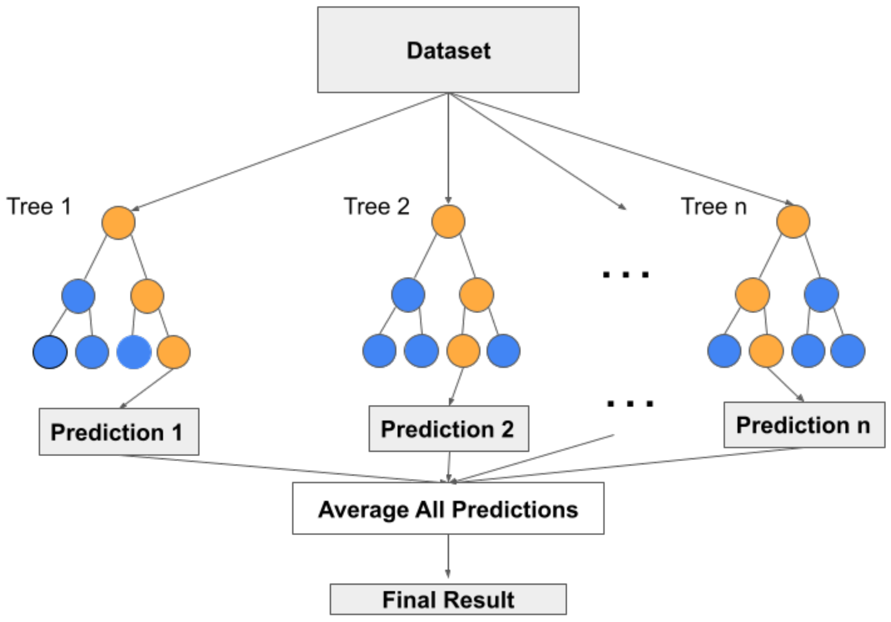

Figure 2.5: Illustration of the making of Random Forest Regressor. Each decision tree's prediction is averaged, and a final result is given. The figure is inspired by Chaya [20]

2.3.4 XGBoost regressor

Similar to Random Forest, XGBoost is an ensemble learning method. Extreme Gradient Boosting is as the name suggests, a gradient-boosted decision tree implementation created to be used with large datasets.

Gradient boosting for regression makes us of decision trees as weak learners. In the beginning, a single leaf is created. For all of the observations in the data set, the leaf represents a guess for the predicted value. The initial assumption made when attempting to predict a continuous number, for instance the column of predicted log2foldchange values, is the average value. It calculates the value of log2foldChange on average. Gradient boost then creates the tree. This tree is based on the errors made by the first tree as shown in figure 2.6. The differences between the observed value and what was predicted are the errors that the previous tree made, called Pseudo Residuals. For the rest of the predicted values, the same approach is used to calculate the residuals [21].

A new tree can then be created using the features from the dataset to predict the residuals. The residuals are replaced by the average at each leaf for corresponding data observation. The original leaf can then be combined with the new tree to make a new prediction for a data observation. This leads to low bias and high variance (overfitting) since the model fits the training data too well. Gradient Boost solves this issue by scaling the contribution from the new tree using a learning rate. It is a value between zero and 1. The new prediction might not be as accurate as the old one, however, it will be better compared to the one created using only the leaf, which is the average value obtained from all observations [21].

Summarised gradient boost builds fixed sized trees based on the previous trees errors. The trees are scaled by the same amount using learning rate. A new tree is then made based on the mistakes made by the previous tree, and scaled. This method of building trees is carried out until the requested number of trees is built or if adding more trees does not result in an improved fit. Doing this, the overall loss will be reduced. The mean squared error (MSE), which is the average squared difference between the predicted and actual values, is the loss function . We want to reduce the regularisation term and the total MSE. L1 and L2 are two regularisation terms in XGBoost Regressor. They are used to add a penalty to the loss function and thereby preventing overfitting. Reducing the regularisation term would create a balance in the model, preventing it from being overfit [22]. The model parameters are updated to lower the overall loss by fitting decision trees to the residuals of earlier trees. Because the gradients are caluculated using the Gradient Descent algorithm in order to minimize loss when adding new models, this method is called gradient boosting [21].

Due to the algorithms speed and the fact that the prediction is not affected by this, XGBoost has become a popular method. In addition to being based on gradient-boosting decision trees, the algorithm has a lot of other features, making it large and complex. Regularization is one of the features applied to a XGBoost model during training.

The use of the weighted quantile sketch approach to deal with sparse data sets is an additional feature of this algorithm. It keeps up with the same level of computational complexity as other algorithms, and at the same time, deals with feature matrices that have non-zero entries. Furthermore, when working with large datasets, it lowers the memory consumption by using cache awareness [23].

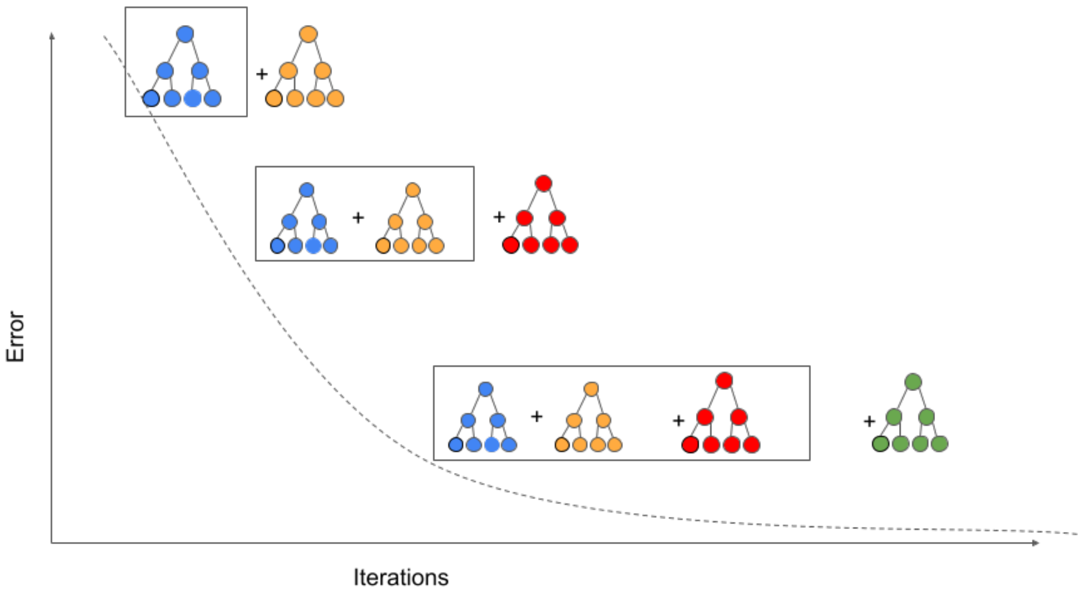

Figure 2.6: Illustration of the gradient boosting method. Adding trees into the model, helps to lower model error. Figure is inspired by Saha [24]

2.3.5 Linear Support Vector Regression

Linear Support Vector Regression (Linear SVR) is a supervised learning algorithm with linear kernel used in regression tasks. In Linear SVR the data points are separated by a linear transformation algorithm. Linear SVR finds a line that best fits the data points. It minimizes the prediction error by approximating the relationship between the input variables and the target variable in the dataset [25]. This algorithm is used for large datasets, and is faster than the SVR algorithm which consider other kernels such as non-linear. [26].

2.3.6 Metrics

To evaluate the performance of the regression models, three metrics were used: R-squared, Root Mean Squared Error and the Pearson correlation coefficient. The root-mean-square error is a measurement of how far away from the true values a model's prediction errors are. R-squared, also known as the coefficient of determination, shows how much of the variance in the dependent variable (in this case log2Foldchange values) in a linear regression model is explained by the independent variables (sequence features). This metric can be used to analyse the pattern of data points in a scatterplot. Pearson correlation coefficient measures the linear relationship between two variables (independent and dependent), and predicted values and true values as well [27]. By dividing the covariance of the two sets by the product of their individual standard deviations, one can calculate the covariance between the two sets.

Equation 2.1 provides the equation for the R-squared metrics. y_k is the true value of the target variable for k-th data point. yˆ_k is the predicted value of the target variable for the k-th data point. y-bar is the mean of the true values of the target variable and n is the number of data points. The equation for RMSE 2.2, y_k, yˆ_k and n stands for the same. For the Pearson correlation coefficient, equation 2.3, x_k is the predicted value of the target variable Y for the k-th data point. y_k is the true value of the target variable Y for the k-th data point. x-bar is the mean of the predicted values of the target variable Y across all data points. y-bar is the mean of the true values of the target variable Y across all data points, and n is the number of data points in the sample.

$$R^2 = 1 - \frac{\sum_{k=1}^{n}(y_k - \hat{y}_k)^2}{\sum_{k=1}^{n}(y_k - \bar{y})^2} \tag{2.1}$$

$$RMSE = \sqrt{\frac{\sum_{k=1}^{n}(y_k - \hat{y}_k)^2}{n}} \tag{2.2}$$

$$r_{X,Y} = \frac{\sum_{k=1}^{n}(x_k - \bar{x})(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{n}(x_k - \bar{x})^2}\sqrt{\sum_{k=1}^{n}(y_k - \bar{y})^2}} \tag{2.3}$$

2.3.7 Related works

There have been many attempts at trying to predict regulatory activity from sequence. The articles by Wang et al. [2] and Bernardo P. de Almeida et al. [28] both employ MPRA methods in their studies.

In Hidra [2], the ATAC-STARR-Seq was used to find chromatin accessible regions. These regions were then analysed to see if there was any regulatory activity. Over seven million accessible DNA fragments were tested in a single experiment. The resulting fragments were used for prediction in a machine learning model called SHARPR-RE.

DeepSTARR is a deep learning model developed in the study by Almeida et al. [28]. It was built to predict enhancer activity from DNA sequences in Drosophila melanogaster S2 cells. To do this research a large dataset of enhancers and corresponding regulatory activity levels was used to train the model. The model learned to identify relevant transcription factor motifs.

Chapter 3

Method

In this chapter we will go into the details of the dataset used to develop the machine learning models for decoding transcriptional regulation in salmon. The implementation of the three machine learning models will also be covered in detail. The programming part of the thesis was done in Python, using Jupyter Notebook as computing platform. Table A.1 in Appendix A, lists each package's use, along with its versions. The source code can be found in the GitHub repository [29], with separate python files for default and tuned models.

3.1 Dataset

The dataset consists of five million training examples (fragments) after filtering, taken from european atlantic salmon liver cells with 29 pairs of chromosomes. To avoid a large number of fragments with few counts, only fragments with a minimum of one count in each sample and an average of at least five (at least 50 in total since there are 10 samples) are chosen.

The file filtered_fragments.fasta, given the variable name "frag" in the Python file, contains the sequence for each fragment that was taken from the reference genome. DESeq has been used to test each fragment to see if the number of observations in the RNA samples and DNA samples differs. The result is saved in DESeq_res.tsv, given the variable "deseq" in the Python file. In one of the columns, we can find "BaseMean" which is the mean of the normalized fragment counts across both the DNA and RNA samples. By using the adjusted p-value column, one can identify differentially expressed genes/fragments with lower chance of false positives using a threshold as 0.05. Fragments below this value are considered significant. Log2FoldChange, which is the estimate size effect, is one of the intriguing columns. Log2Foldchange is chosen as the target variabel, i.e the value the machine learning model will predict. It is the counts of RNA over counts of DNA. Since the dataset includes a small number of highly expressed DNA sequences, we use Log2FoldChange = 0 as a cutoff to identify both up and down-regulated genes. A positive log fold change indicates that the RNA is being up-regulated, and a negative log fold change that it is being down-regulated (more DNA than RNA). In other words, around 0, where there has been little change, are the fragments are lowly expressed. Those that are very significant have a considerable difference in log2fc from 0, i.e., are either highly positive or highly negative and are differentially expressed fragments.

3.2 Preprocessing

To be able to do feature engineering, one has to make sure that the given raw dataset is clean and ready to use. Figure 3.1 illustrates the preprocessing steps for each model.

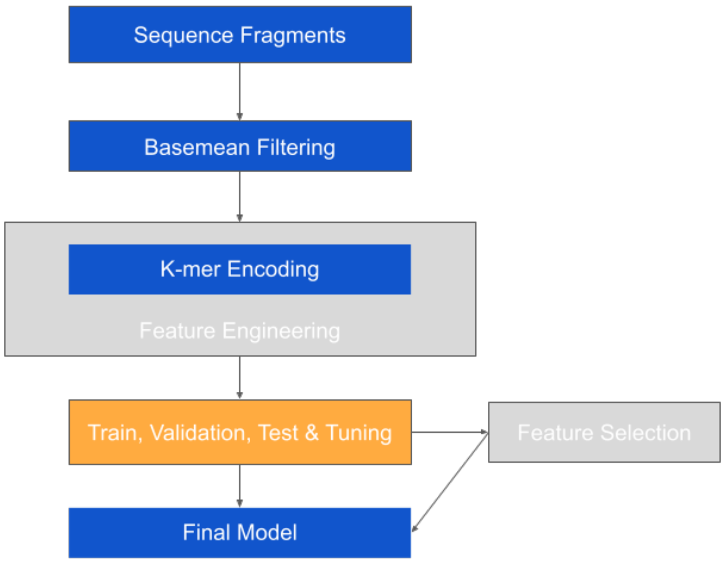

Figure 3.1: Preprocessing steps for every model

3.2.1 Data Cleaning

As previously mentioned, the dataset consists of over five million fragments. Five of these fragments were found to have ”n” nucleotides, which means that the nucleotides were unspecified and thus represents unknown data. It could either be adenine, guanine, cytosine or thymine. In case they would interfere with the machine learning models, the fragments that contained ”n” nucleotides were removed from the dataset. Keeping many ”n” could result in the model making the wrong prediction for a Log2FoldChange value and cause bias or noise in the model.

3.2.2 Splitting the dataset

The dataset was split into two separate tests, a training set and a test set. Fragments from chromosome 21 and 25 were extracted after cleaning the data, and set aside as test set. The training set includes fragments from all chromosomes except chromosome 21 and 25, used for training the machine learning model. We want the model to generalize well to solve unseen data. For instance, only using the training data could lead to the model just learning a specific pattern really well and fail to accurately predict new data. Splitting the dataset can therefore be useful for discovering overfitting in the model.

3.2.3 Filtering Basemean

Basemean is the mean of the normalized fragment counts across both the DNA and RNA samples. In order to improve the performance of the machine learning models, we wanted to leave out some of the fragments that could make the prediction of transcriptional regulation uncertain. Previous predictions without basemean filtering showed that there were noise in the data which lead to machine learning models not able to predict well. The data set was evaluated with various percentages of top basemean by testing parts of the data that have high basemean values. Fragments with top 10% basemean were chosen and applied to training, validation and test sets.

3.2.4 Feature Engineering

Feature engineering is an important field in machine learning. It is used prior to the training of the model step and involves extracting useful and important features from the raw dataset. By transforming the model one can achieve a model that can learn the data better, which then can improve the performance of the machine learning model. The underlying goal in this case is to train the machine learning models for decoding transcriptional regulation.

The DNA sequences must first be encoded before we can use them as input in the machine learning model. The DNA sequences must be transformed because they are made up of nucleotides. Machine learning algorithms do not work with text data, therefore one has to transform the sequences to numerical data in a way that the algorithm can understand.

The features that were created from the dataset were k-mers. K-mer encoding is an easy way for analyzing sequence data. They are helpful when it comes to matching sequences. A k-mer is a length subsequence of k characters from a biological sequence. 2-mers, 3-mers, 4-mers, 5-mers and 6-mers were calculated and stacked together in a dataframe. Figure 3.2 shows an example of how k-mer features are created. Based on the chosen length, in thise case k = 4, the sequence CACACAT is divided into k-mers of length 4 that overlap. Using Scikit-learn's CountVectorizer, each k-mer is converted to a matrix with corresponding frequency counts. As visualized, each column contains a single k-mer feature, and each row represents a sequence fragment. The k-mer sizes were chosen based on the fact that the dataset is particularly large with five million samples and the resources were limited. Choosing large k-mer sizes could potentially result in longer training time.

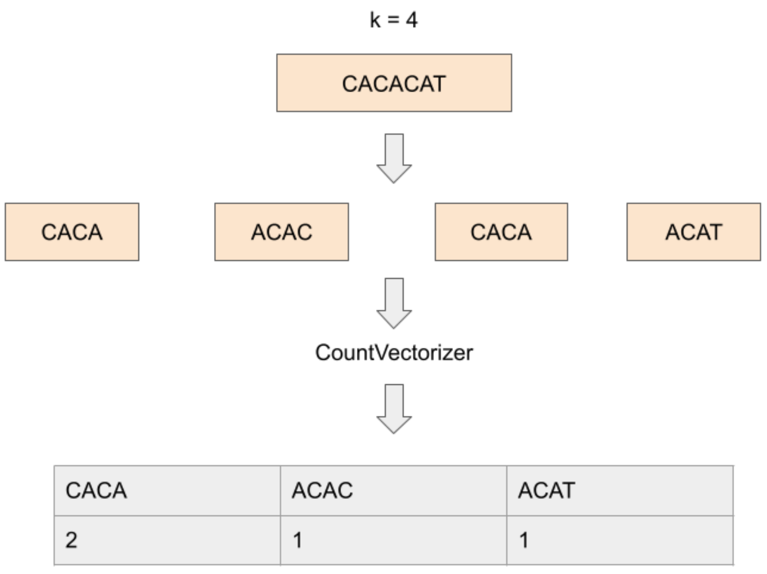

Figure 3.2:

3.2.5 Train, Validation and Test

The training set was further split into a new training set and a validation set. The new training set was used for training the machine learning model, and the validation set was used to evaluate the performance of the model during the tuning of hyperparameters. By splitting the datasets, we make sure that the model generalises well to unseen data , which brings us to the test set. The dimensions of the training, validation and test sets are given in table 3.1.

Dataset	X data shape	y data shape
Train set	(393813, 5456)	(393813,)
Validation set	(168778, 5456)	(168778,)
Test set	(20696, 5456)	(20696,)

Table 3.1: Dimension (shape) of training, validation and test set

3.3 XGBoost Regressor Model

The XGBoost llibrary was imported from the open source library "XGBoost" [30], which offers an effective implementation of gradient boosting algorithms. The XGBRegressor algorithm was imported from the library to be used in Scikit-learn. In Chapter 2, section 2.3.3 the algorithm is described in detail. After the preprocessing steps, the first XGBoost Regressor Model was implemented using default parameters from Scikit-learn library version 1.2.0.

3.3.1 Hyperparametertuning

After creating the default model with XGBoost regressor, we wanted to see whether we could obtain an improved model by searching for ideal parameters, in the process known as hyperparameter tuning. Machine learning algorithms on its own can produce good results, however, they might not necessarily achieve the highest possible accuracy. When performing hyperparameter tuning, the machine will explore and choose the best parameters for a model. Note that hyperparameters differs from model parameters. Hyperparameters are chosen before training a model and controls how the model is structured, whereas model parameters are selected by training data [31]. Hyperparameter optimization methods such as grid search and randomized search can be used to search for optimal parameters. We will focus on the latter approach in this thesis. The following list contains the hyperparameters that were tuned.

- **gamma**: the minimum loss reduction to create new tree-split
- **learning_rate**: to avoid overfitting, step size reduction is used in updates
- **max_depth**: maximum tree depth
- **n_estimators**: maximum number of boosting rounds

RandomizedSearchCV and GridSearchCV

RandomizedSearchCV and GridSearchCV are both implemented in Scikit-learn. To identify the best parameter values, RandomizedSearchCV chooses a fixed number of random parameter combinations from among all possible combinations in a specified grid. K-Fold cross validation, explained in section 2.3 is performed for each combination of values. In other words, a different combination of the parameters will be selected by RandomizedSearchCV for each iteration. In contrast, GridSerachCV tests all parameter combinations. Because of its speed, RandomizedSearchCV is considered more advantageous over GridSearchCV. Randomized search was chosen for the XGBoost Regressor model based on the fact that the dataset provided for this thesis is large. It was performed with 10 iterations and 3-fold cross validation.

3.3.2 Feature importance

The trained XGBoost Regressor model and the default Random Forest Regressor model were used to extract feature importance scores. Feature importance is an attribute in Scikit-learns tree based models. The scores shows the importance for each feature when it comes to predicting

Table 3.2: Hypermarameters tuned in RandomizedSearchCV with repspective values and the chosen parameters for XGBoost Regressor tuned model

Hyperparameter	**Values**	**Selected**
Gamma	uniform(0, 0.5)	0.03
Learning rate	uniform(0.03, 0.3)	0.29
Maximum depth	randint(2, 6)	5
Number of estimators	randint(100, 150)	139

the Log2FoldChange value. In the next step, the most important features were selected for predicting transcriptional regulation. These sequences were then written in to a fasta file.

3.3.3 Motif search

After extracting the important sequence features from the trained model, Tomtom was used to search for known motifs. Tomtom is a a motif comparison tool used to evaluate motifs in comparison to a database of identified motifs [32]. A motif is sequence pattern which in this case can be associated with transcription factor binding sites. Based on the chosen transcription factor database, Tomtom will look for matches to each of the motifs. Given that the k-mer features created range from 2 to 6, the lengths of the sequence/motif that were extracted have the same range.

3.4 Random Forest Regressor Model

Theory for this algorithm can be found in chapter 2, section 2.3.2. The Scikit-learn library version 1.2.0 was used to implement this model.

Following the preprocessing steps described in 3, a Random Forest Regressor model was created with the default parameters from the Scikit-learn library. The model was fitted using the training set. Then the separate test set was used to evaluate the performance of the model. Later on, during the hyperparametertuning stage, the validation set was applied. Randomized search was selected for the this model as well, using ten iterations and three-fold cross validation.

3.4.1 Hyperparametertuning

The following list contains a set of hyperparameters that were inspired by Koehrsen [17]:

- **n_estimators**: number of trees in the forest
- **max_depth**: maximum number of levels in each tree
- **max_features**: maximum number of features that are considered for dividing a node
- **bootstrap**: method for sampling data points

Table 3.3: Hyperparameters tuned in RandomizedSearchCV with respective values and the chosen parameters for Random Forest Regressor tuned model

Hyperparameter	**Values**	**Selected**
Number of estimators	300, 500, 600,700, 900	900
Maximum depth	4, 6, 8, 12, 14, 16	14
Maximum features	auto, sqrt	auto
Bootstrap	True, False	True

3.5 Linear SVR

For the Linear SVR model, the preprocessing was done the same. In contrast to the to other models, Linear SVR was tuned using GridSearchCV with ten iterations and three-fold cross validation.

3.5.1 Hyperparametertuning

- **C**: the regularization parameter
- **Epsilon**: specifies a tolerance margin where errors are not penalized

Table 3.4: Hyperparameters tuned in GridSearchCV with respective values and the chosen parameters for Linear SVR tuned model

Hyperparameter	**Values**	**Selected**
C	0.001, 0.01, 0.1, 1, 10, 100, 1000	10
Epsilon	0.001, 0.01, 0.1, 1, 10, 100	0.1

Chapter 4

Results

In this chapter, results from the three machine learning models are presented. Each model's performance is shown in a table using the three metrics: : R-squared, Root Mean Squared Error and the Pearson correlation coefficient. The results from the extracted sequence features that were used to determine association with known transcription factor sites will be presented.

4.1 XGBoost Regressor Default VS Tuned

A comparison between the default model and the tuned model will be made in this section. The default model was created using the default parameters implemented in Scikit-learn.

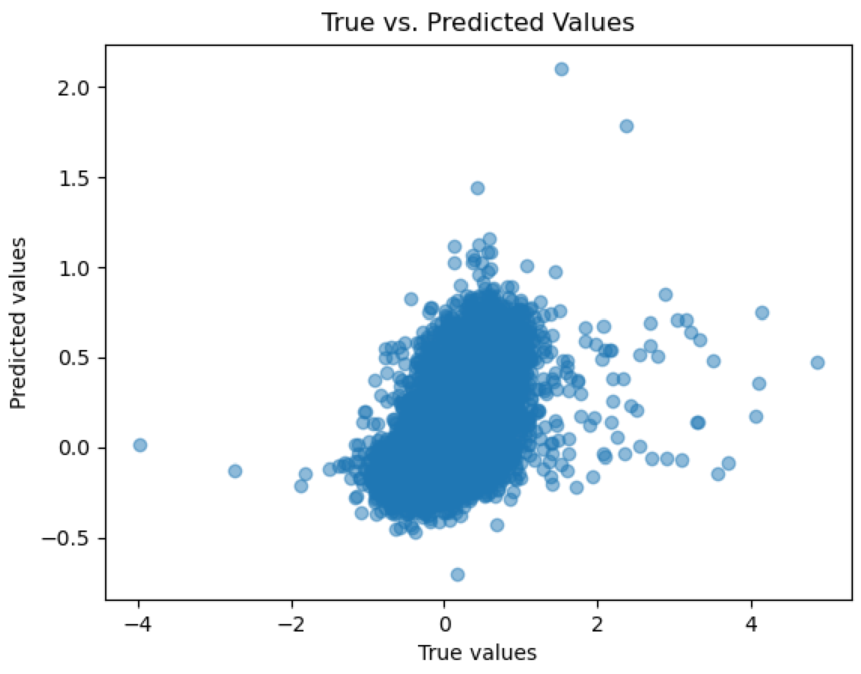

Figure 4.1: Default XGB Regressor

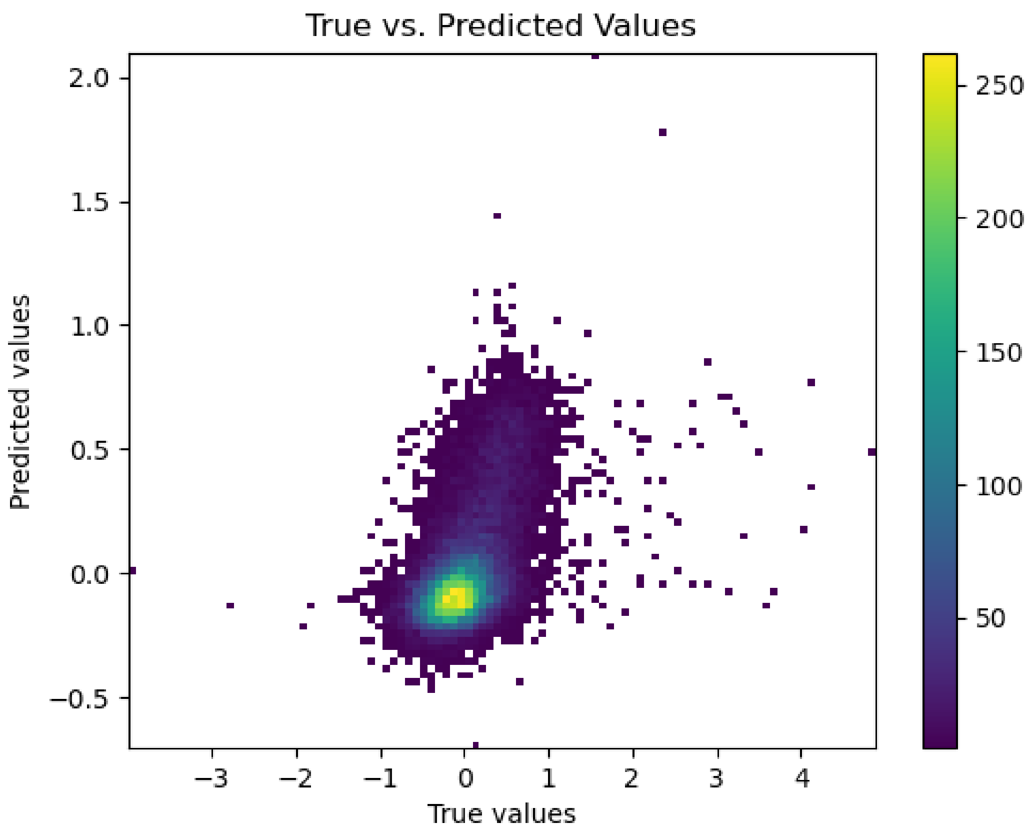

Figure 4.2: Default XGB Regressor

The scatterplot shows a visual representation of the degree to which the model's predictions correspond to the true values. The estimated Pearson correlation coefficient, which measures how strongly and in which direction two variables are correlated linearly is listed in the metrics, along with r-squared and root-mean-square error described in theory 2.3.6. It gives a single number as the score. By using a scatterplot one can explore the relationship between the two variables. For instance, discover a linear pattern that indicates a positive relationship between the predicted and true values.

The model's Pearson correlation coefficient of 0.53 shows that it achieved a moderate positive strength of association. The plot shows that the relationship is not as strong as we had anticipated. The machine learning model is poor at predicting the genes that are significantly up- and down-regulated. Several points go towards 0, as we can see from the large number of observations in the middle in figure 4.2. When certain values are highly expressed, they are classified as lowly expressed. We are aware that in the data set, little expressed genes outnumber differentially expressed genes. Machine learning is strongest where it has examples. However, it predicts better at 2 than at -2 which indicates that there are very upregulated genes in the dataset, hence the model is better at predicting upregulation than downregulation.

Table 4.1: Metrics performance for default XGBoost Regressor Model

Metric	**Score**
R-squared	0.27
Root Mean Squared Error	0.49
Pearson Correlation coefficient	0.527

Next, the results from the tuned XGBoost Regressor Model will be presented. As explained earlier, four hyperparameters were tuned to find the optimal parameters for the model. The scatterplot (fig 4.4) shows the relationship between the predicted values and the true values for the tuned XGBoost Regressor model.

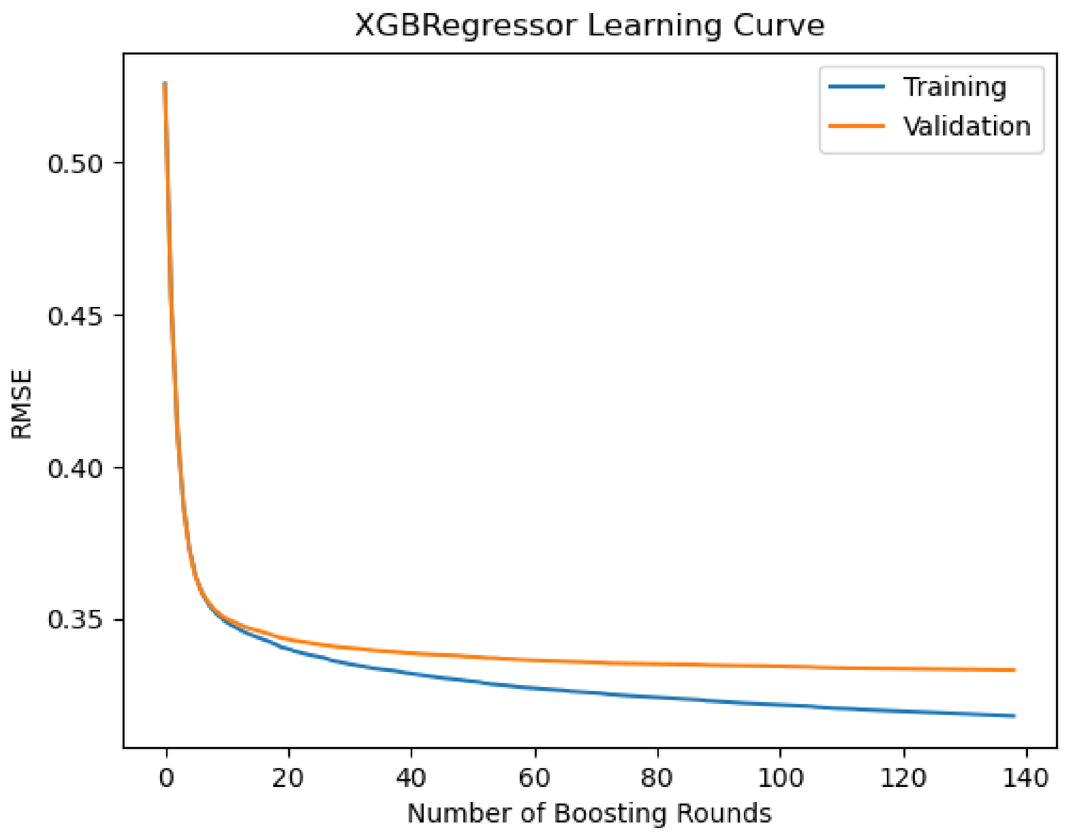

Figure 4.3: XGB Regressor Learning Curve

The y-axis in figure 4.3 displays the root mean squared error for both the training and validation datasets, and the x-axis displays the number of boosting rounds (number of estimators or trees).

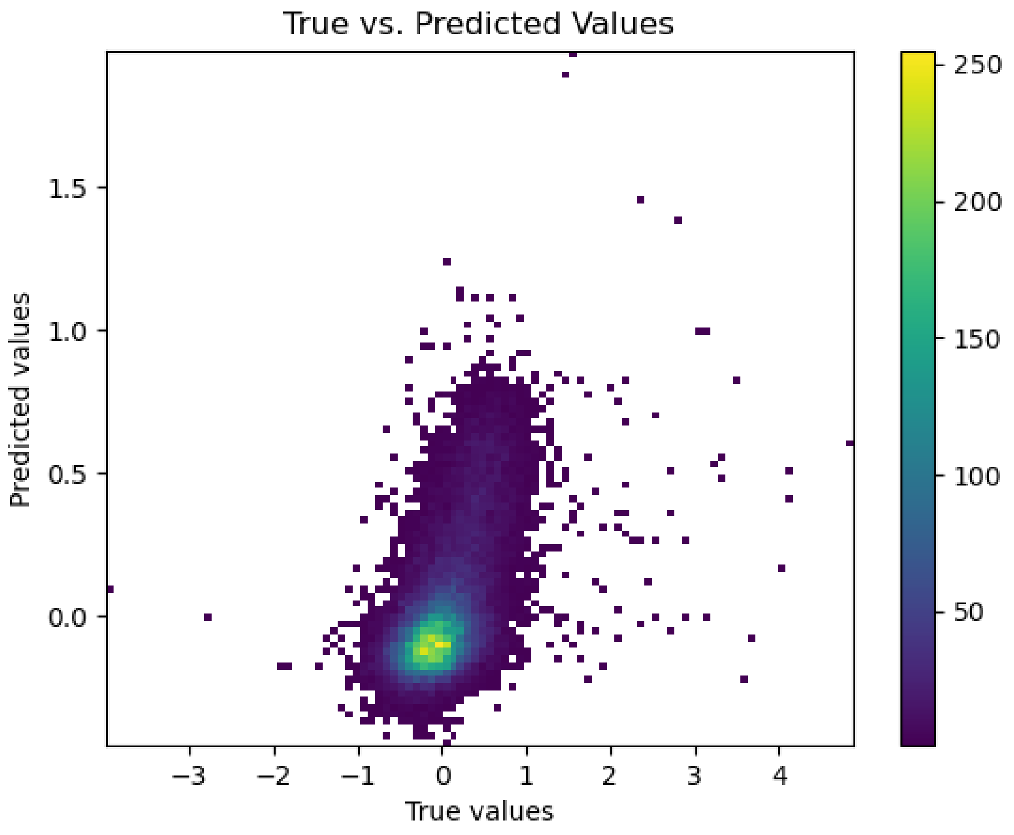

Figure 4.4: The tuned XGB Regressor scatterplot

As seen from 4.4, the choice of hyperparameters did not significantly improve the model. Comparing the metrics for the default model in table 4.1 and the tuned model in table 4.2, we can see that there is not much difference in the scores. From the learning curve, we can see from the beginning that neither of the graphs are overfitting or underfitting. The validation curve starts to flatten a bit at around 139 estimators, and increasing the number of boosting rounds to the model may result in better improvements. The r-squared score of 0.28 shows that the model explains only a small portion of the variance in the log2Foldchange values, based on the sequence features in the model. As a result, a large amount of the variation in the log2Foldchange values remains unknown. However, the data points in the plot are little bit more spread, predicting certain values accurately. For instance, the model gets better at predicting points that are upregulated of value 2.

Table 4.2: Metrics performance for tuned XGBoost Regressor Model

Metric	**Score**
R-squared	0.28
Root Mean Squared Error	0.49
Pearson Correlation coefficient	0.53

4.2 Random Forest Regressor: Default VS Tuned

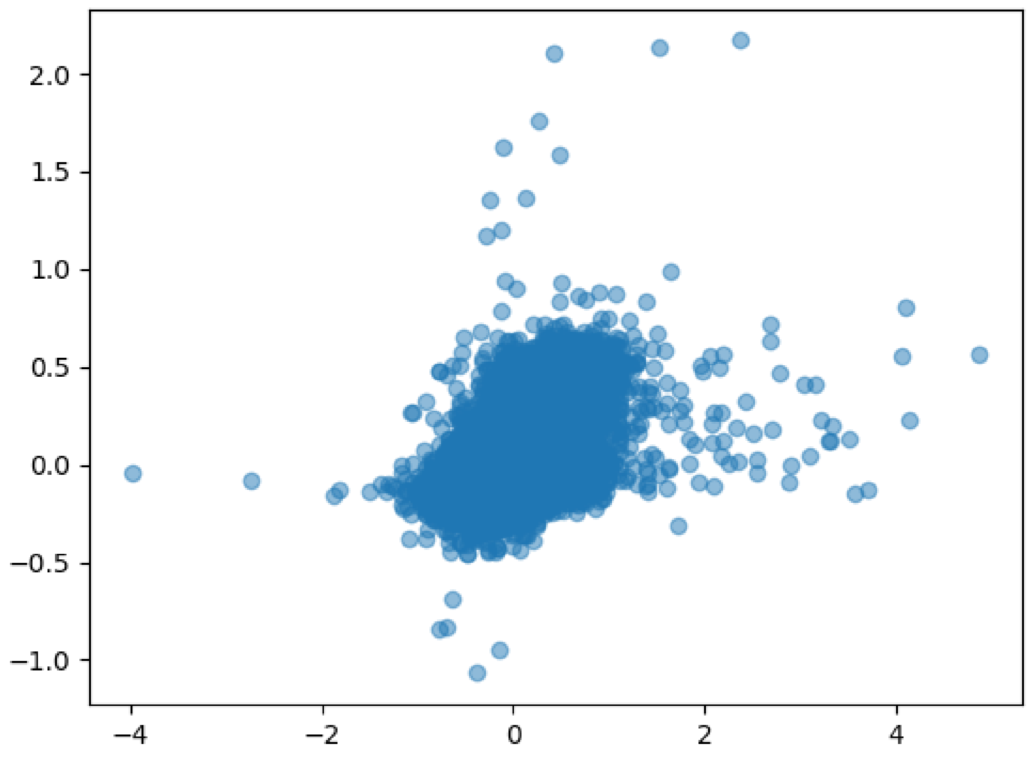

Figure 4.5: Default Random Forest Regressor scatterplot

Based on the previous figures from XGBoost Regressor and figure 4.5, we can observe a continuous pattern. The scatterplots suggests a non-linear relationship between the predicted and true values. From figure 4.5, we can see that the model is not always able to predict the values accurately. However, it is able to predict some log2Foldchange values from 1 to 2 in this model as well. The R-squared score of 0.26, as can be seen from table 4.3, indicates that there is a lot of variance that is not fully explained by the model.

Table 4.3: Metrics performance for default Random Forest Model

Metric	**Score**
R-squared	0.26
Root Mean Squared Error	0.49
Pearson Correlation coefficient	0.52

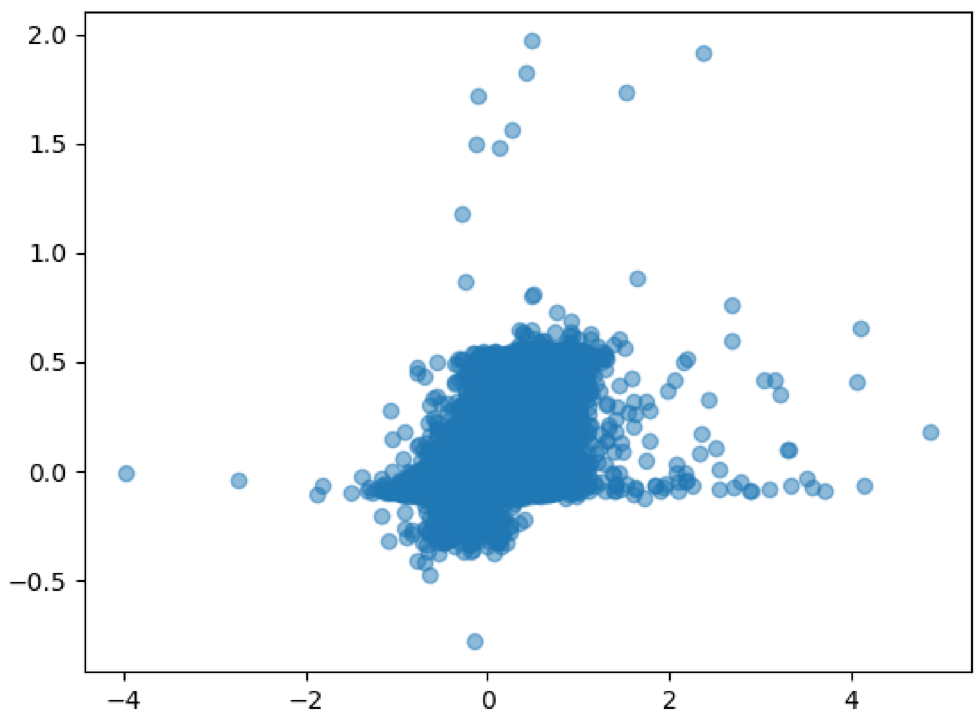

Figure 4.6: Tuned Random Forest Regressor model

The tuned Random Forest Regressor model, however, performs slight worse then the default model. Comparing the R-squared score for the default model from table 4.3 and the tuned model in table 4.4 , shows that in terms of explaining the variance in the log2Foldchange values, the default model may perform better. The difference between the two scores for both models is still small. Even with the results of the Pearson Correlation coefficient, there is not much of a difference between the two scores. Choosing the best model should therefore be thoroughly evaluated. However, more highly expressed values appear to be classified as lowly expressed on the tuned Random Forest Regressor plot, indicating poor performance.

Table 4.4: Metrics performance for tuned Random Forest Model

Metric	**Score**
R-squared	0.23
Pearson Correlation coefficient	0.48

4.3 Linear SVR: Default VS Tuned

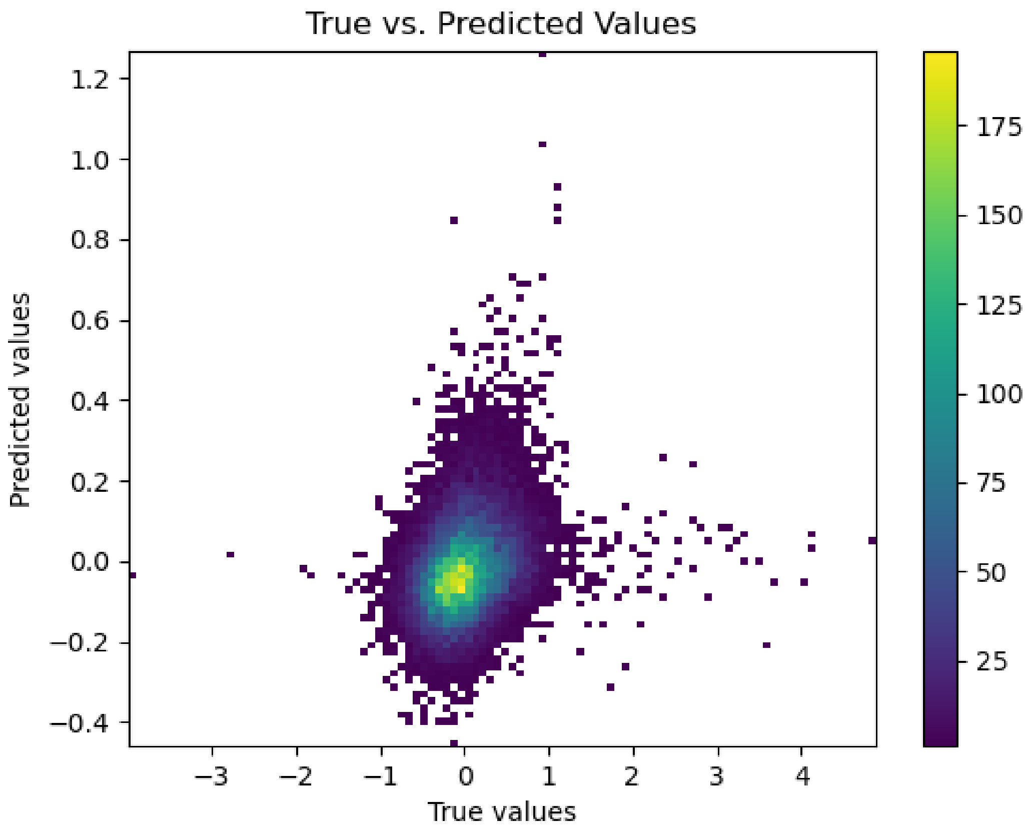

Figure 4.7: Default Linear SVR scatterplot

Table 4.5: Metrics performance for default Linear SVR

Metric	**Score**
R-squared	0.07
Root Mean Squared Error	0.52
Pearson Correlation coefficient	0.279

Comparing the two Linear SVR models, we can observe that it is not a huge difference between the scores and the scatterplots. Both models have achieved a low R-squared score. The best results that can be achieved from the hyperparametertuning is an R-squared score of 0.08 and a Pearson Correlation of 0.29.

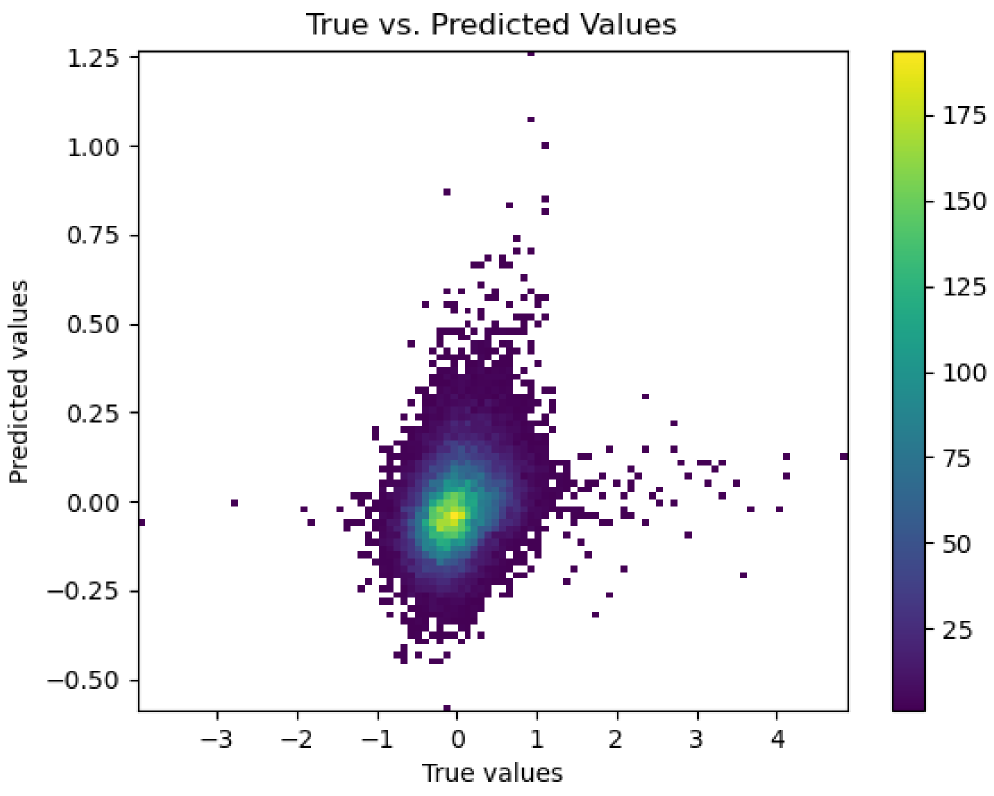

Figure 4.8: The tuned Linear SVR scatterplot

Table 4.6: Metrics performance for tuned Linear SVR

Metric	**Score**
R-squared	0.08
Root Mean Squared Error	0.52
Pearson Correlation coefficient	0.288

4.4 Motif Search Discovery

A variety of interactions between many transcription factors are involved in the process of transcriptional regulation. The motif search has been centered on discovering transcription factors related to liver because the sequence fragments originate from liver cells, also known as hepatocytes. The JASPAR Vertebrates and UniProbe mouse database was used to compare the motifs. The motif search yielded many significant motifs. A few of them that are related to hepatocytes are presented in figure 4.9 and figure 4.10.

The motif match plot, which is created by Tomtom following a motif search, shows how closely related the query motif and the motifs in the target database are to one another. The position of the motif in the query sequence is shown on the x-axis, and the similarity score between the query motif and the motifs in the database is shown on the y-axis. The similarity between the matches is measured by bits, which are a logarithmic measure of a match's significance. Higher bit scores indicate more significant matches.

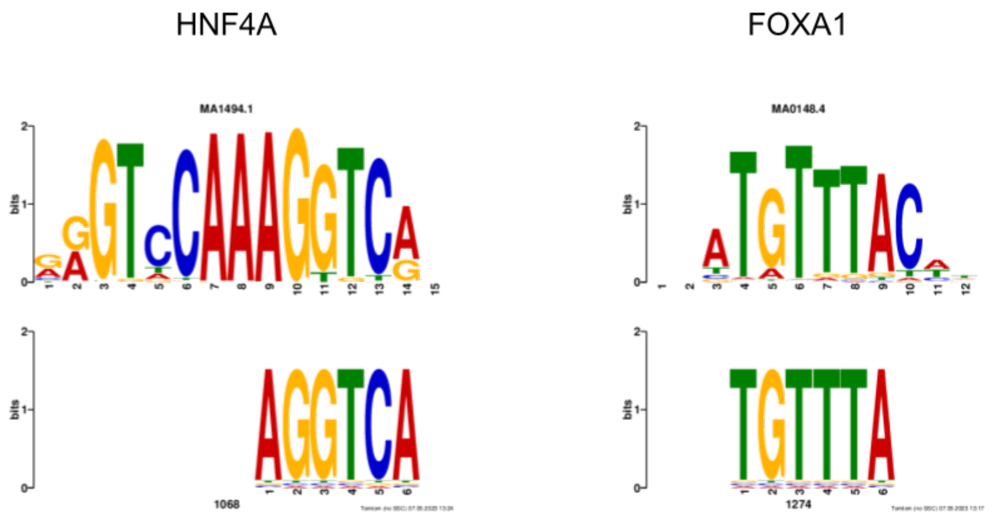

Figure 4.9: Transcription factors HNF4A and FOXA1

Hepatocyte nuclear factor-4-alpha (HNF4A) motif presented in figure 4.9 belongs to the nuclear receptor family of transcription factors. HNF4A binds to specific DNA regions and helps regulating the expression of certain genes. This transcription factor is more present in liver cells than any other transcription factor. It is crucial in the development and function of specific tissues and organs in the body. A large number of the actively transcribed hepatic genes are regulated by HNF4A [33].

FOXA1 (figure 4.9) belongs to the forkhead class of DNA-binding proteins. It is known as hepatocyte nuclear factor 3-alpha. FOXA1 makes it possible for other proteins to enter the compacted chromatin, and thereby serves as a pioneer factor [34]. In addition to interacting with chromatin, the transcription factor function as a transcriptional activator for liver-specific transcripts. The creation of tissue-specific gene expression and the control of gene expression in differentiated tissues such as liver cells, are all regulated by this transcription factor [35].

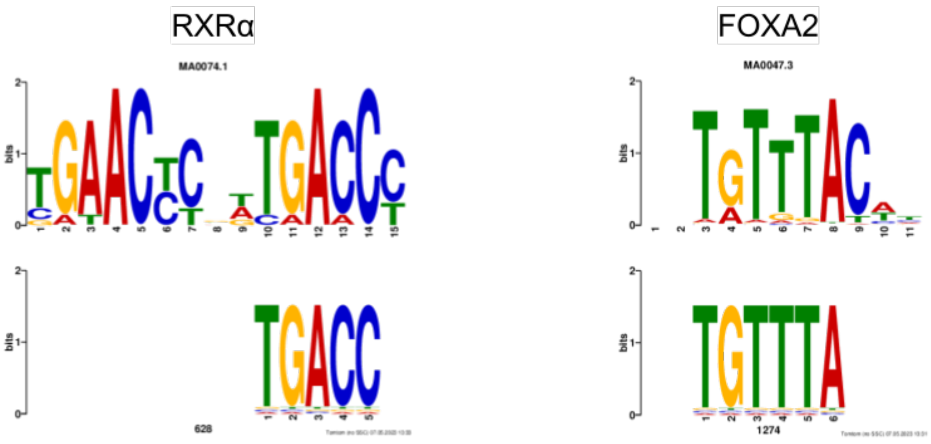

Figure 4.10: Transcription factors RXRalpha and FOXA2

The Retinoid x reseptor alpha (RXRalpha) transcription factor in figure 4.10 is expressed in liver cells and is cruical for the function of other nuclear reseptors. It is involved in the regulation of gene expression related to liver and the functioning of the liver itself [36]. Forkhead box protein A2 (FOXA2) (figure 4.10), known as hepatocyte nuclear factor 3-beta, is involved in the regulation of gene expression and the control of metabolic processes in the liver, as well as other tissues [37].

Chapter 5

Discussion

In this chapter we will discuss the data quality and the performance of the three machine learning models.

5.1 Data Quality

The machine learning models ability to make predictions can be significantly affected by the dataset used in this thesis. Looking back at the results in chapter 4, we can observe that the majority of the predicted log2FoldChange values in the scatterplots show low levels of transcriptional regulation.

The genomic data was generated using the salmon genome. Genetic variations may have an impact on how well the machine learning models perform. The sequences from the reference genome in ATAC-STARR-Seq may have varied from the fish that were sampled. As a result, the expression of some fragments can be affected by certain variations. For instance, the accessibility of DNA sequences to transcription factors and other regulatory regions can be affected by genetic variations, thereby affecting how the genes are expressed. Genetic variations can also have an impact on how strongly transcription factors bind to DNA, which again can have an affect on gene expression.

Another factor that can influence the expression levels and the prediction of the models is the DNA sequencing method used in this thesis. The machine learning models performance may be affected by the ATAC-STARR-seq method. Sequencing errors can bring in noise to the data. In order to guarantee the accuracy of the sequence fragments data, quality control methods should be used. If the quality of the data is not taken into consideration, machine learning models may make predictions that are incorrect.

By selecting sequence fragments within the top 10 % basemean from the dataset, we wanted to focus on capturing the fragments that are more certain for prediction, making it easier to predict and improve the model. As the machine learning models could learn more effectively using top 10 % basemean fragments, this filtering method seemed to be beneficial. We have many fragments in the experiment and thus many counts, which is why this supports our thesis. However, the model performances showed that this technique still resulted in mostly moderate correlation and low prediction power given the results of the machine learning models.

5.2 Model Performance

In Chapter 4 the results from the default model and tuned model of XGBoost Regressor, Random Forest Regressor and Linear SVR were presented. Although the XGBoost Regressor models and the default Random Regressor model generated results that were remarkably similar, the XGBoost regressor got marginally better results.

The XGBoost Regressor model performed quite similar for both the default and tuned model as seen in table 4.1 and table 4.2. The reason for the similar performance can be because the hyperparameter tuning was only done with 100 to 150 estimators, which may seem like a small number for a training set with 300 000 samples and 5000 features. Increasing the number of estimators could potentially improve the performance of the model. The number of estimators were chosen based on the fact that training many trees resulted in longer training time and could be computationally expensive. However, the time needed to implement the default model and the tuned model with 100 to 150 estimators was not that long. Given that the learning curve in figure 4.3 neither did indicate overfitting or underfitting, the selected estimator of 139 seemed reasonable with possibility of further tuning.

Feature selection using feature importance to select important features, which is not presented in the results, was applied to the tuned XGBoost Regressor model. It yielded the best performance using all features, and was therefore not presented in the results section. We could observe that the performance dropped as a result of features getting removed. The more features we used, the better correlation, R-squared score and RMSE score we got. This confirms that all the features are important for the prediction of the Log2Foldchange values for this particular model. Feature selection was not experimented on the Random Forest Regressor model and Linear SVR due to time limitations.

Compared to the XGBoost Regressor model, implementing the Random Forest Regressor models was quite time consuming. Tuning the Random Forest Regressor models would become one of the most challenging parts of this thesis. Considering that Random Forest Regressor combines several decision trees to make predictions and given the size of the training set and, it was expected that the tuned model would be time consuming to develop. From the results in subsection 4.2 we could see that the tuned model did not outperform the default model. This might be because during the tuning process, RandomizedSearchCV's search space could have been too limited, such that the combined parameters could not surpass the default model. For instance, the maximum depth range in the hyperparameter could not have enough variation to affect the performance of the model. Another reason might be that the number of iterations, ten in this case, was too small. An additional explanation could be that the model is overfitting. The default model might have been the best fit for this task.

The default Random Forest Regressor model performed relatively similarly to the XGBoost Regressor models, being able to capture some complex relationships. Although the models only had marginal difference in the metrics performances, there were some differences in the prediction of the log2FoldChange values that could be observed in the scatterplots. This indicates that the models are making different errors. The reason for this could be that the models are sensitive to different k-mer features in the data.

In terms of performance, the ensemble methods are more comparable than Linear SVR. Linear SVR was shown to perform poorer than the other algorithms. One could observe a big drop in metrics performance for this algorithm. The cause of this could be that the algorithm assumes a linear relationship between the sequence fragments (k-mer features) and the log2FoldChange values. In contrast to Linear SVR, both Random Forest Regressor and XGBoost regressor algorithms are able to tackle non-linear trends. Previously it was mentioned that the scatterplot figures suggests a non-linear relationship between the predicted and true values. Still, the Linear SVR model is able to make some predictions correct, based on figure 4.8 and metrics performances. When it comes to training time, the Linear SVR models ran as fast as the default XGBoost Regressor model.

All plots showed a consistent pattern, however there may be additional patterns that the model fails to capture. Every evaluated model improved only marginally when tuned, perhaps this is the best result they can achieve based on the basemean filtering and the dataset. Despite the moderate results and k-mer features from range two to six, which resulted in short motifs, it was possible to extract sequence features that drive transcription and associate them with known transcription factors related to liver, such as HNF4, FOX1, FOX2 and RXRalpha.

5.3 Future Work

Due to limitations in time, several things were not explored. Scikit-learn's CountVectorizer was used to convert k-mers to a matrix with frequency counts. Only 2-mers, 3-mers, 4-mers, 5-mers and 6-mers were chosen because of the large dataset and limited resources. CountVectorizer is computationally expensive as the size of the feature space increases with the length of k-mers, resulting in longer training time. The extracted sequence motifs from the machine learning models were initially thought as too short to be used in the motif search to find significant motifs. It could be of interest to create features with longer k-mer sizes for association with transcription factor binding site without going to through the process of longer training time.

It could also be of interest to explore the parameters in hyperparametertuning for the models a bit further. Especially the Random Forest Regressor, which was not explored enough because of time limitations. Furthermore, one can think about the practical use of what has been done. For instance using basemean to filter fragments. For datasets with high basemean fragments, this can be effective. It is clear that there is still much work to be done in this field.

Chapter 6

Conclusion

In this thesis, sequence fragments from an ATAC-STARR-seq experiment are used to predict regulatory activity. The ATAC-STARR-seq experiment has been done on a salmon genome. The dataset consists of over five million examples. Fragments within top 10% basemean are selected to improve the machine learning model. The data is split into three sets: training, validation and test sets. In the feature engineering process, k-mer features of sizes two to six are created from the dataset.

Three classical machine learning regression methods are used to predict transcriptional regulation from the sequence fragments. The models that are implemented are XGBoost Regressor, Random Forest Regressor and Linear Support Vector Regression. XGBoost Regressor and Random Forest Regressor are selected because of their ability to perform well and identify complex trends in data. Linear SVR is chosen due to its qualities as a fast running regression algorithm, and works effectively when handling large datasets.

Based on the metrics performances and the visualization of the relationship between the true and predicted log2foldchange values, we can conclude that the XGBoost regressor models and the Random Forest Regressor models achieved moderate correlation score with low predictive power. Linear SVR achieved a weaker correlation score with less predictive power than the other models. Despite the results, it is possible to extract sequence features from the trained machine learning models and associate them with known transcription factors related to hepatocytes. The discoveries made in this thesis can be used as a foundation for further exploration directed at parameter tuning and feature engineering with larger k-mers for prediction of trancriptional regulation in salmon.

Bibliography

[1] European Bioinformatics Institute. *Epigenetic Profiling.* Accessed April 2, 2023. URL: https://www.ebi.ac.uk/training/online/courses/functional-genomics-i-introduction-and-design/common-study-types-in-functional-genomics/epigenetic-profiling/.

[2] Xiaoyang Wang et al. "High-resolution genome-wide functional dissection of transcriptional regulatory regions and nucleotides in human". In: *Nature Communications* 9.1 (2018), p. 5380. DOI: 10.1038/s41467-018-07746-1.

[3] Madeleine Price Ball. "Chemical structure of DNA". In: (). URL: https://commons.wikimedia.org/w/index.php?curid=1848174.

[4] Nature Publishing Group. *Protein Function.* Accessed April 1, 2023. n.d. URL: https://www.nature.com/scitable/topicpage/protein-function-14123348/#:~:text=Proteins%20are%20responsible%20for%20nearly,waste%20cleanup%2C%20and%20routine%20maintenance..

[5] Atdbio. *DNA, RNA and protein synthesis.* Accessed: 2022-11-14. 2021. URL: https://atdbio.com/nucleic-acids-book/Transcription-Translation-and-Replication.

[6] Khan Academy. *Overview: Eukaryotic gene regulation.* Accessed April 1, 2023. URL: https://www.khanacademy.org/science/ap-biology/gene-expression-and-regulation/regulation-of-gene-expression-and-cell-specialization/a/overview-of-eukaryotic-gene-regulation.

[7] Paul P. Liu. *Chromatin.* Accessed April 1, 2023. URL: https://www.genome.gov/genetics-glossary/Chromatin#:~:text=Chromatin%20refers%20to%20a%20mixture,fit%20in%20the%20cell%20nucleus..

[8] Theresa Phillips. "Regulation of transcription and gene expression in eukaryotes". In: *Nature Education* 1.1 (2008), p. 199. URL: https://www.nature.com/scitable/topicpage/regulation-of-transcription-and-gene-expression-in-1086/.

[9] Julie Segre. *Promoter.* Accessed April 11, 2023. URL: https://www.genome.gov/genetics-glossary/Promoter.

[10] Khan Academy. *Transcription factors.* Accessed April 3, 2023. URL: https://www.khanacademy.org/science/ap-biology/gene-expression-and-regulation/regulation-of-gene-expression-and-cell-specialization/a/eukaryotic-transcription-factors.

[11] Bernstein0275. *Regulation of transcription in metazoans (animals).* Wikimedia Commons. URL: https://commons.wikimedia.org/wiki/File:Regulation_of_transcription_in_metazoans_(animals).jpg (visited on 05/02/2023).

[12] Bioinformatics Home. *DESeq.* URL: https://bioinformaticshome.com/tools/rna-seq/descriptions/DESeq.html#gsc.tab=0.

[13] Google Cloud. *Artificial Intelligence vs. Machine Learning.* URL: https://cloud.google.com/learn/artificial-intelligence-vs-machine-learning.

[14] IBM. *Machine Learning.* URL: https://www.ibm.com/topics/machine-learning.

[15] University of York. *What is Reinforcement Learning?* URL: https://online.york.ac.uk/what-is-reinforcement-learning/.

[16] BMC Blogs. *What is Bias-Variance Tradeoff in Machine Learning?* 2021. URL: https://www.bmc.com/blogs/bias-variance-machine-learning/.

[17] Will Koehrsen. *Hyperparameter Tuning the Random Forest in Python Using Scikit-Learn.* 2018. URL: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74.

[18] Gurucharan M K. *Machine Learning Basics: Decision Tree Regression.* 2020. URL: https://towardsdatascience.com/machine-learning-basics-decision-tree-regression-1d73ea003fda.

[19] IBM. *Random Forest - IBM.* URL: https://www.ibm.com/topics/random-forest.

[20] Chaya. *Random Forest Regression.* 2019. URL: https://levelup.gitconnected.com/random-forest-regression-209c0f354c84.

[21] Josh Starmer. *Gradient Boost Part 1 (of 4): Regression Main Ideas.* 2019. URL: https://www.youtube.com/watch?v=3CC4N4z3GJc.

[22] Um Albert. *L1/L2 Regularization in XGBoost Regression.* 2021. URL: https://albertum.medium.com/l1-l2-regularization-in-xgboost-regression-7b2db08a59e0.

[23] Simplilearn. *What is XGBoost Algorithm in Machine Learning?* 2023. URL: https://www.simplilearn.com/what-is-xgboost-algorithm-in-machine-learning-article.

[24] Sumit Saha. *XGBoost vs LightGBM: How Are They Different.* 2023. URL: https://neptune.ai/blog/xgboost-vs-lightgbm.

[25] Analytics Vidhya. *Support Vector Regression Tutorial for Machine Learning.* 2020. URL: https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/.

[26] Ashwin Raj. *Unlocking the true power of Support Vector Regression.* 2020. URL: https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0.

[27] Laerd Statistics. *Pearson Product-Moment Correlation.* URL: https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php.

[28] Bernardo P. de Almeida et al. "DeepSTARR predicts enhancer activity from DNA sequence and enables the de novo design of synthetic enhancers". In: *Nature Genetics* 54.5 (2022), pp. 613–624. DOI: 10.1038/s41588-022-01048-5.

[29] Harini Jeyakumar. *Data Science Thesis.* 2023. URL: https://github.com/harinjey/thesis-data-science.

[30] XGBoost. *XGBoost.* URL: https://xgboost.readthedocs.io/en/latest/install.html.

[31] Analytics Vidhya. *A Comprehensive Guide on Hyperparameter Tuning.* 2022. URL: https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-machine-learning-hyperparameter-tuning/.

[32] Tomtom. *TOMTOM: Tool for Motif Matching.* URL: https://meme-suite.org/meme/tools/tomtom.

[33] Omim. *HEPATOCYTE NUCLEAR FACTOR 4-ALPHA; HNF4A*. Online Mendelian Inheritance in Man. 2021. URL: https://www.omim.org/entry/600281#6.

[34] Wikipedia. *FOXA1*. 2023. URL: https://en.wikipedia.org/wiki/FOXA1.

[35] UniProt Consortium. *P55317 FOXA1$_H$UMAN*. URL: https://www.uniprot.org/uniprot/P55317.

[36] Genecards. *RXRA Gene - Retinoid X Receptor Alpha*. 2023. URL: https://www.genecards.org/cgi-bin/carddisp.pl?gene=RXRA.

[37] Wikipedia. *FOXA2*. 2022. URL: https://en.wikipedia.org/wiki/FOXA2.

Appendix A

Table of Python libraries

Python	Version	Application
”Matplotlib”	3.6.2	Plotting
”Pandas”	1.5.2	Preprocessing
”Numpy”	1.23.5	Array operations
”Sklearn”	1.2.0	Implementation of XGBoost Regressor, Random Forest Regressor and Linear SVR
”SciPy”	1.10.1	Generating numbers for tuning

Table A.1: Python packages used in this study