UNIVERSITY PARIS-SACLAY
MASTER'S THESIS

Machine Learning approaches to improve the CRISPR/Cas9 system in *Drosophila melanogaster*

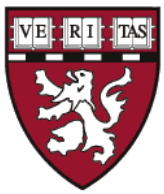
Author:
Pierre MERCKAERT

Supervisors:
Dr. Claire Yanhui HU
Dr. Stephanie MOHR
Dr. Raghuvir VISWANATHA

A thesis submitted in fulfillment of the requirements for the degree of Master's of Bioinformatics

in the

DRSC of the Perrimon Lab
Genetics Department
Harvard Medical School



HARVARD
MEDICAL SCHOOL

Declaration of Authorship

I, Pierre MERCKAERT, declare that this thesis titled, "Machine Learning approaches to improve the CRISPR/Cas9 system in *Drosophila melanogaster* " and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UNIVERSITY PARIS-SACLAY

Abstract

Harvard Medical School
Genetics Department

Master's of Bioinformatics

Machine Learning approaches to improve the CRISPR/Cas9 system in *Drosophila melanogaster*

by Pierre MERCKAERT

In this internship project the goal is to enhance the Drosophila RNAi Screening Center (DRSC) pipeline for CRISPR sgRNA design by improving the sgRNA efficiency predictions. To do so I use Machine Learning algorithms to model the efficiency of sgRNAs targeting essential genes, based on their characteristics, from a *Drosophila melanogaster* genome-wide pooled CRISPR screen. The Machine Learning models that I identify and optimize are implemented in a command line tool with which single or batch sgRNA efficiency predictions can be made. I show in this work that, for sgRNAs designed for the *Drosophila* genome, my program has better performance than the tool currently used in the DRSC, as well as Azimuth, the current sgRNA efficiency prediction state-of-the-art. Thus, the program developed and described here gives accurate CRISPR sgRNA efficiency predictions and is ready to be used by the *Drosophila* community.

Acknowledgements

I want to warmly thank my mentors Stephanie Mohr, Claire Hu, Ram Viswanatha, and my PI Norbert Perrimon for granting me this life-changing opportunity, providing me with such a welcoming work environment and guiding me all throughout this internship.

I am thankful for the entire Bioinformatics team, Claire, Aram, Verena, Dove, as well as all the techs, post-docs and researchers for their precious help, feedback, kindness and the passionating discussions.

I am immensely grateful for the invaluable connections and friendships I built during my stay. Eric, Gary, Hanna, Benjamin, Limmond, Felipe, Laure, Alex, Ricky, Cooper, Shannon, Jimmy, Daniel, and many more, thank you all for your unending joy, your support, advice, spicy memes and all the memories together that I deeply cherish.

I also want to thank Sabrina Shrestha and Catherine King for their very precious and efficient help to facilitate setting up this internship and guide me by providing all the information I needed.

Finally, I thank the University Paris-Saclay and my professors, Olivier Lespinet, Christine Froidevaux and Daniel Gautheret, for the excellent formation I received, the quality of their teachings and the grant they helped me obtain.

Thank you all !

Contents

List of Figures

Chapter 1

Introduction

This work took place in the Bioinformatics team of the Drosophila RNAi Screening Center (DRSC) of the Perrimon Lab at the Genetics Department of Harvard Medical School, in Boston Massachusetts, USA. The bioinformatics team of the DRSC grew from the need for state-of-the-art reagent design and screen data analysis approaches. The team has since expanded its efforts to 6 members now (me included) in response to community needs, providing support for additional types of reagents, tools, analysis, visualization, and integration.

1.1 CRISPR/Cas9

1.1.1 The technology and its limits

The CRISPR/Cas9 system has attracted significant attention in recent years due to its potential capacity to create designer edits in any genome as well as its ease of use. In particular, it holds promise unraveling the genetic basis of human disease as well as for curing genetic diseases, as demonstrated through correction of a variety of disease-causing mutations in human cultured cells and animal models [1, 2, 3]. The term "genome editing" refers to technologies which can produce genome modifications, such as mutagenesis or indels, at specific sites in the genome of living organisms. Genome editing relies on the production of site-specific double-strand DNA breaks (DSBs) and the subsequent endogenous repair through the error-prone non-homologous end-joining (NHEJ) or the error-free homology-directed repair (HDR) pathways. Due to its error-prone nature, NHEJ repair often leads to DNA sequence alterations at the targeted DSB sites. If the mutation occurs in an exon, NHEJ can lead to a complete gene knock-out, as indels introduced in the DNA sequence can lead to frameshifts in the DNA sequence, which can lead to missense or nonsense mutations [4].

Prior to CRISPR/Cas9, Zinc-Finger Nucleases (ZFN) and Transcription Activator-Like Effector Nucleases (TALEN), in which sequence-specific nuclease domains are multiplexed in arrays to achieve greater target specificity, were demonstrated to be capable of introducing designer edits into genomes [5]. The reason that CRISPR/Cas9 represents a significant advance over these prior technologies is that it uses short, encodable RNA sequences (single guide or sgRNAs) to guide the nuclease to target sites.
Briefly, Cas9, the trans-activating crRNA (tracrRNA) and a CRISPR RNA (crRNA), composed of a 20 nucleotides long sgRNA complementary to the targeted region, and the tracrRNA, are designed and expressed in the desired cell.

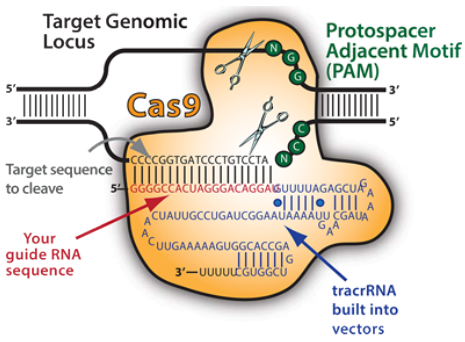


FIGURE 1.1: Structure of the CRISPR/-Cas9 system

Cas9 is activated by binding to the crRNA-tracrRNA complex, thus forming the CRISPR/Cas9 active complex. Using the sgRNA template, Cas9 "scans" the genome and creates a DSB at the specific region complementary to the sgRNA. It is important to note that Cas9 will not successfully bind to or cleave the targeted DNA sequence if it is not followed by a correct Protospacer Adjacent Motif (PAM) sequence. The PAM is a 3 base pair DNA sequence, 5'-NGG-3' for wild-type S.p.Cas9, immediately following the DNA sequence targeted by the Cas9 endonuclease.
Because all DNA editing technologies have the potential to cause both desired (on-target) edits and undesired (off-target) edits, a significant challenge is to identify reagents for each locus of each genome that maximize on-target activity while minimizing off-target activity. The ease of cloning and expressing CRISPR sgRNAs allows massively parallel hypothesis testing, allowing the user to simultaneously test hundreds or thousands of RNA sequences to achieve finer and finer precision over target site editing. An important step forward into this area has been recent attempts to identify the "rules" that correlate with successful sgRNAs and the challenges associated with poorly performing sgRNAs. This work focuses on optimizing sgRNA design to improve on-target efficiency in *Drosophila melanogaster (D.mel)*.

1.1.2 CRISPR/Cas9 in *Drosophila melanogaster* and the DRSC

D.mel was among the first organisms used for genetic analysis, and today it is one of the most widely used and genetically best-known of all eukaryotic organisms. It is a powerful model organism for biological research because it has only four pairs of chromosomes, breeds quickly, lays many eggs, has visible genetic markers, has a fully sequenced genome and has about 60% of its genes conserved with Human [Wikipedia : `https://bit.ly/2meNlIs`].
D.mel cell lines are also widely used for genetic screening. S2 cells in particular have previously shown to be a potent cell line to perform with RNAi screening [6, 7], as well as more recently, drug targets or essential genes identification with CRISPR/Cas9 screens [8, 9].
The Drosophila RNAi Screening Center (DRSC) at Harvard Medical School aims to facilitate *Drosophila* cell-based RNAi and CRISPR screens, and support related functional genomics projects, in the broadest possible research community. The quality of the CRISPR screens is directly correlated to the accuracy and efficiency of the sgRNAs used. Thus, this intership aims at using high-end computational technology, such as Machine Learning, to improve sgRNA design for CRISPR screens in the DRSC.

1.1.3 CRISPR genetic screens and sgRNA design rules uncovered

Many sgRNA design rules have been uncovered thanks to large CRISPR/Cas9 screens in human cell-lines: Using truncated sgRNA of 17 to 19 nucleotides, instead of the standard 20 nucleotides, may improve sgRNA specificity [10]; Lengthening the tracrRNA can improve sgRNA efficiency [11]; Mismatches in the seed region (12 PAM-proximal nucleotides) greatly lowers cleavage efficiency [12]; sgRNAs with very high or low GC content are less effective and sgRNAs targeting the last coding exon have lower gene knock-out rates than those targeting earlier exons [13]. It is important to note that these screens were conducted in human cells so the results may not be applicable to invertebrates. In fact, *D.mel* sgRNA design algorithms show little correlation with human design algorithms even when applied to the same templates (Claire Hu *unpublished*, [14]). Moreover, many position matrices, based on the nucleotide preference at each position of the sgRNA, have been designed both in mammals and *D.mel*. Consistent results between studies show a strong preference for Cytosine and low preference for Thymine at position 18, and a strong preference for Guanine and low preference for Thymine at position 20 [15, 16, 17].

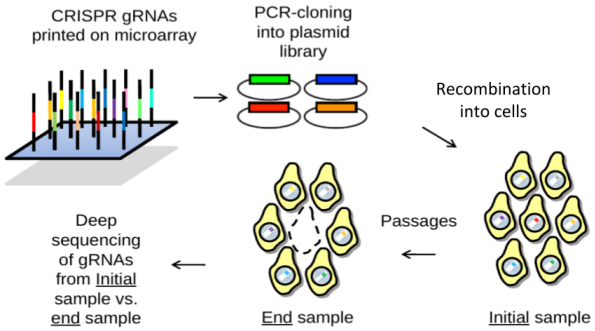


FIGURE 1.2: genome-wide pooled CRISPR screen

This particular result is consistent with CRISPR screens conducted in my thesis lab on *D.mel* [8, 9]. Due to its off-target potential and the irreversible, and somewhat random, nature of DNA repair, CRISPR is not currently suitable for therapeutic application (except for extreme cases). However, CRISPR ease of use is fully leveraged through pooled genetic screens in cells where off-target effects are not of crucial importance and specific edits can be selected through phenotype screening. My internship project is based on a genome-wide pooled CRISPR screen conducted by Viswanatha et al on S2R+ *D.mel* cells [9]. S2R+ *D.mel* cells derive from S2 cells and express Wnt receptors on the cell surface. Briefly, the library of sgRNAs (and Cas9) is cloned into plasmids that are recombined into cells. After the desired amount of cell passages (or time), a selection screen is performed (e.g. drugs or growth), then the gRNAs from the selected cells are sequenced and their read count is compared to the reference pre-passages.

1.2 Machine Learning

Another popular method for improving CRISPR sgRNA design is Machine Learning of efficient sgRNAs' characteristics to quickly predict the efficiency of untested sgRNAs. Machine Learning (ML) is a branch of artificial intelligence that gives systems the ability to learn automatically and improve themselves without being explicitly programmed. ML can provide a potent way of discovering new important features for sgRNA design as the powerful statistical algorithms available can detect subtle patterns in the input data distribution. But most importantly, ML models are able to make predictions by fitting a statistical model to the distribution of a response variable of interest according to the values of its explanatory variables. For CRISPR sgRNA design, this means that ML algorithms can use the sgRNA characteristics of an 'unseen' sgRNA (not used to build the ML model) to predict the expected efficiency of that sgRNA.
I used two methods of prediction: Regression and Classification. For regression, the response variable (sgRNA efficiency) is based on the log2FC and is thus continuous. For classification, the response variable is binary: 1 or 0 based on the log2FC of the top and the worst 20% sgRNA respectively. Regression models try to find the best relationship between the response variable (sgRNA efficiency) and the data distribution (sgRNA characteristics). Classification models try to find the best separation in the response variable categories (effective or ineffective sgRNA) according to the data.
Several tools are available to design and predict CRISPR sgRNA efficiency, such as Synthego's CRISPR Design Tool, CRISPOR [18], the Broad Institute's sgRNA Designer or Microsoft Research's CRISPR guide design tool. These tools all use the same state of the art machine learning models in terms of CRISPR sgRNA on-target efficiency prediction [19, 20]. Even though most of these tools implement sgRNA design in the *D.mel* genome, the prediction models used to assess sgRNA efficiency were built based on mammalian cell CRISPR screens, so the efficiency predictions may not be relevant for invertebrate sgRNAs. Hence my internship project.

1.3 Internship Goals

1. **Identifying sgRNAs targeting essential genes from a genome-wide CRISPR screen in *D.mel* cultured cells.**
   sgRNAs targeting essential genes cause the cell's death and are thus easily identifiable by sequencing since their amount in the cell is supposed to dramatically drop (because the cells with such sgRNAs die). Hence, based on an unbiased genome-wide pooled CRISPR screen conducted on *D.mel* in my thesis lab [9], we planned to identify essential genes using two popular software tools, BAGEL and MAGeCK [21, 22]. The sgRNAs targeting these essential genes were then retrieved to calculate their log2 fold change as this will be the metric assessing gene knock-out efficiency.

2. **Identifying and optimizing the best ML model to predict sgRNA efficiency in *D.mel*, and compare it to the state of the art regarding sgRNA efficiency prediction.**
   After extracting relevant features from the sgRNAs, based on the literature for ML prediction of on-target efficiency [19, 20], multiple relevant regression and classification ML models are trained, tested, validated, and the best ones compared to current on-target efficiency prediction tools.

3. **Develop a tool to predict *D.mel* sgRNA efficiency, that can be integrated to my thesis lab's pipeline for CRISPR design.**
   The tool should be fast, easy to use and understand, able to process multiple types of sgRNA inputs and outputs, and integrable in the Drosophila RNAi Screening Center (DRSC) Bioinformatics' pipeline for CRISPR design.

Chapter 2

Material and Methods

2.1 Identification of sgRNA targeting essential genes

The following tools and protocols were used to identify sgRNAs targeting essential genes from the genome-wide pooled CRISPR screen conducted on *D.mel* S2R+ cells [9].

2.1.1 DRSC Integrative Ortholog Prediction Tool (DIOPT) - Version 7.1

https://www.flyrnai.org/cgi-bin/DRSC_orthologs.pl
DIOPT is a powerful online tool for identifying orthologs among Human and model organisms [23]. DIOPT lets users find ortholog pairs for a specified gene or genes identified by one or many of published ortholog prediction approaches.

2.1.2 Model-based Analysis of Genome-wide CRISPR-Cas9 Knockout (MAGeCK) - Version 0.5.7

https://sourceforge.net/p/mageck/wiki/Home/
MAGeCK is a computational tool developed from the recent genome-scale CRISPR-Cas9 knockout screens (GeCKO) technology [22]. MAGeCK is able to identify essential genes based only on the read counts from genome-wide CRISPR screens. The Maximum Likelihood Estimate option was used to compute the z-score for each gene from their read counts. The z-score is the number of standard deviations a gene read count is from the population mean. A negative z-score of *-N* means that the read count for this gene, post CRISPR experiment, is lower than the mean read count of all genes by *N* standard deviations. Thus, a gene with a low z-score correlates with gene essentiality.

2.1.3 Bayesian Analysis of Gene Essentiality (BAGEL) - Version 0.91

https://sourceforge.net/p/bagel-for-knockout-screens/wiki/Home/
BAGEL is a Bayesian classifier for pooled library genetic perturbation screens [21]. It uses training sets of known essential and nonessential genes to estimate what the fold change distribution of an essential or nonessential gene should look like. Based on 3 CRISPR or RNAi genome-wide screens, 3 different lists of essential and non essential genes were given to BAGEL as training sets : The "RNAi" list is based on a *D.mel* RNAi screen by Boutros et al. [24]; The "bj" list is based on a *Human* RNAi screen by Björklund et al. [25]; And the "Hart" list is based on a *Human* CRISPR screen with the Toronto Knock-Out library by Hart et al. [26]. Then, for each uncharacterized gene, BAGEL takes all observations of sgRNAs targeting that gene and makes a probabilistic statement about whether those observations are more likely

to be sgRNAs targeting an essential or a non essential gene. A Bayes Factor for each gene is reported. Positive Bayes Factors indicate confidence that the gene is essential. DIOPT was used to retrieve the Flybase ID (FBgn) of the genes from the gene names of the "bj" list. Since the "Hart" list is a list of Human genes, DIOPT was used to find orthologs in *D.mel* and retrieve their FBgn.
Regarding the "RNAi" list, since the RNAi screen from Boutros et al. has 5 replicates, I chose to focus on the replicate that would have the most BAGEL-predicted essential genes based on a False Discovery Rate analysis of highly expressed genes. See [Appendix A.1] and next section for details. The essential genes training list input for BAGEL was, for each of the 5 replicates, the top 500 genes with the lowest z-score and a log(FPKM+0.01) > 1 (high expression level). The non-essential genes training list input for BAGEL was, for each of the 5 replicates, the top 500 genes with the smallest FPKM > 1 (very low expression). I focused on the first replicate since it is the one with the most BAGEL-predicted essential genes.

2.1.4 Data processing and analysis

Raw read counts from the pooled CRISPR screen were median normalized before using BAGEL or computing the log2 Fold Change (log2FC) for MAGeCK. The normalization was conducted as follows :

1. The 10% genes with the lowest read counts in the reference were removed.
2. The reference and post-CRISPR experiment read counts were logarithmized (base e).
3. Each log read count row was averaged. Genes with a null average log read count were filtered out.
4. The average log read count was subtracted from the natural log reference and post-CRISPR read counts. Then the exponential of the medians were calculated for both.
5. The original initial and final read counts were divided by their respective medians.

Once normalized, the sgRNA read counts are either inputed as is to compute the MAGeCK z-score for each gene, or the log2 Fold-Change (log2FC) is calculated and inputed alongside the essential/non-essential genes training list to compute the BAGEL BF score for each gene. The Log2 Fold-Change is a metric describing how much a quantity changes going from an initial to a final value. For a CRISPR screen, if the log2FC of the sgRNA is negative it means that the amount of sgRNA at the initial timepoint (reference) is higher than the amount of sgRNA at the final timepoint (t0 + 45 days, in our case).
Then, the genes are sorted according to their z-score or BF, from the "most essential" to the "least essential", and the gene expression False Discovery Rate (FDR) is cumulatively calculated. We set a 5% FDR threshold to extract with confidence highly expressed genes that are predicted to be essential.
The False Discovery Rate (FDR) for each sample of predicted essential genes is computed by analyzing the Fragments Per Kilobase of transcript per Million mapped reads (FPKM) of each gene. The FPKM is a measure of gene expression normalized for sequencing depth and gene length. Since essential genes have a high expression level [9], the null hypothesis is that all essential genes found with BAGEL or MAGeCK have a high FPKM. We cumulatively add bins of 5 genes and calculate the rate of essential genes with low FPKM (FDR). We set a threshold at 5% FDR to confidently select the best sample of essential genes.

2.2 CRISPR sgRNA efficiency predictions

2.2.1 Machine Learning models development

Data processing. The log2FC for each of the 7514 sgRNA targeting the 1340 essential genes identified were normalized between 0 and 1, and used as the response variable to be predicted by regression models. For classification models the response variable is binary, so the top 20% sgRNAs with the lowest log2FC and the bottom 20% sgRNAs with the highest log2FC were respectively considered as effective and non effective sgRNAs.
Predictive models. I used the following statistical models for regression : (i) linear regression, (ii) support vector machine regression, (iii) ridge regression, (iv) gradient-boosted regression tree and (v) random forest. For classification models I used : (vi) logistic regression, (vii) linear support vector classifier, (viii) ridge classifier, (ix) decision tree classifier, (x) gradient-boosting classifier and (xi) random forest classifier. These models were selected based on ease of implementation and usage in the literature, they were implemented in python with the *scikit-learn* (v0.19.2) library.
There are 2 types of ML tasks: Supervised and Unsupervised. Briefly, supervised learning is performed using a ground truth, meaning that we have prior knowledge of what the output values of our data should be (e.g. sgRNA efficiency). Unsupervised learning, does not have labeled outputs, so its goal is to infer the natural structure present within a set of data points (e.g. TSNE of RNASeq).
Featurization. For reference, 23mer sgRNA stands for the 20mer + the 5'-NGG-3' PAM, and 30mer sgRNA stands for the 23mer sgRNA + context sequence, namely NNNN[20mer sgRNA]NGGNNN. The continuous features used to predict sgRNA efficiency were : The amount of each possible 4 nucleotides and 16 dinucleotides in the entire sgRNA (position independent order 1 and order 2), the GC content of the 20mer sgRNA, the Melting Temperatures for the start (positions 1-7), middle (8-15), end (16-20) and full 20, 23 or 30mer sgRNA. Continuous features were standard scaled so that they are centered around 0 and have variance of the same order. If a feature has a variance that is significantly larger than other features', this feature might dominate the ML function and make ML algorithm unable to learn from other features correctly as expected.
The binary features used to predict sgRNA efficiency were : for each position in the sgRNA, the presence or absence of all the possible combination of 4 nucleotides (120 for a 30mer) or dinucleotides (464 for a 30mer). The features are referred as position dependent order 1 and order 2. As well as the presence or absence of the 16 possible combinations of the 2 nucleotides relative to the PAM "NGGN" (unused feature for 20 and 23mer sgRNA predictive models). This type of featurization is called one-hot encoding, meaning that among all the possible nucleotide combination for a given position in the sgRNA, only one of them is "hot" or "on" or 1, all the others are 0. For example, position 1 of the 30mer can take A/T/G/C, so the position 1 feature gets converted into 4 features, one for each nucleotide (A1, T1, G1, C1), but only one of these 4 features can be "hot" or 1 for any given sgRNA. This type of encoding gives great precision but also largely increases the amount of features in the model, which may not be a good thing, hence the need for selection of the most relevant features.
To sum up, the ML models take into consideration 625, 469 and 409 features for efficiency prediction of 30, 23 and 20mer sgRNA respectively.
Performance evaluation and cross-validation. The goal of a good ML model is to generalize well the training data to any data. This allows us to make predictions in the future on data the model has never seen. The two biggest causes for poor performance in ML are underfitting and overfitting. Underfitting refers to a model that can neither fit the training data nor generalize to new data. Overfitting refers to a model that models the training data too well and is unable to generalize to new data that differs from the training data. Ideally, we want to select a model at the sweet spot between underfitting and overfitting, able to fit well the training data (but not too well) and to generalize to new data. To do so, we try to

optimize the ML model performance metrics, such as how well the model fits the training data and how accurate are the model predictions on new data.
For regression, several metrics are used to assess the performance of the model. The coefficient of determination R^2 (R squared) shows the percentage of variation in the response variable that is explained by the explanatory variables. A R^2 of 1 means, in our case, that all of the variation in sgRNA efficiency is explained by the sgRNA's characteristics, and that the model fits perfectly our data, which is not a desirable thing, in order to prevent overfitting. The R^2 can be negative if the model fit is worse than the basic $y = x$ regression. However, R^2 never decreases as it assumes that even the addition of a somewhat useless feature to the model explains the variation of the response variable. The adjusted-R^2 takes into account the number of features considered by the ML model an penalizes for features that do not fit the model. Thus, the adjusted-R^2 is always smaller than the R^2. The Root Mean Square Error (RMSE) is used to assess the differences between the values predicted by a model and the "true" values observed. We want the predictions to be as close as possible to the observed values used to train the ML model, so a smaller RMSE is better than a higher one.
For classification, the Area Under the Curve (AUC) is used to assess model performance. The AUC is the accuracy metric of classification models and is calculated based on the Receiver Operating Characteristic (ROC) curve of the model. The ROC curve is a plot of the True Positive Rate (TPR) (i.e. 1 predicted as 1 and 0 as 0) against the False Positive Rate (FPR) (i.e. 1 predicted as 0 and 0 as 1). The closer the graph is to the top and left-hand borders, the more accurate the test. Likewise, the closer the graph to the diagonal, the less accurate the test. A test with AUC = 0.5 is a useless test. Thus, the greater the area under the curve (AUC), the more accurate the test.
10-fold cross-validation (CV) was always performed when assessing the performance for feature selection, hyperparameter optimization or prediction accuracy. One round of 10-fold of cross-validation involves partitioning the data into 10 subsets, training the model on 9 of these subsets (called the training set), and testing the model on the remaining subset (called the testing set). This process occurs 10 times so that all the 10 subsets are used once as testing set. Then the testing results are averaged over the rounds to give an estimate of the model's fitting or predictive performance.
A train/test approach is used to assess the prediction performance of the models on data that has not been used to build the models. Indeed, we want the sgRNA efficiency predictions to be generalizable to sgRNAs that the model has not "seen". The training set consists of the 7514 sgRNAs targeting essential genes. The test sets consist, for *D.mel*, of 1030 sgRNAs chosen from the the pool of 79283 sgRNAs of the genome-wide screen so that the dataset has a "somewhat" (less sgRNAs available for extreme efficiency values) equally distributed "true" efficiencies along the percentiles (i.e. as many sgRNAs with an efficiency of [0.3-0.4] as sgRNAs with an efficiency of [0.6-0.7] etc...), so that the efficiency distribution of the test set is as less skewed as possible. To asses predictions performance on *Human* sgRNAs, the test set consists of 2600 sgRNAs selected from the Doench et al. CRISPR screen [20], also in a manner to equally distribute efficiency along the percentiles.
Feature selection. Feature selection is common practice in ML in order to simplify the models by removing redundant or irrelevant features. The scikit-learn function for Recursive feature elimination and cross-validated selection of the best number of features (RFECV) was used to perform feature selection. Given a linear ML model, the goal is to select features by recursively considering smaller and smaller sets of features. First, the model is trained (with default hyperparameters) on the initial set of features, the model performance is calculated (R2 or AUC) and the importance of each feature is obtained either through a "coef" or a "feature importance" attribute. Then the least important features are pruned from the current set of features. That procedure is recursively repeated on the pruned set until the *desired* number of features to select is eventually reached [scikit-learn API]. This *desired* number of features is found using cross-validation : For every step where *n* number of features are eliminated, the performance

score is calculated. The number of features left at the step which gives the maximum score is considered to be the best number of features for your data.
Hyperparameter optimization. The parameters of the models used were tuned to optimize their performance. An exhaustive cross-validated grid search (GridSearchCV) of the parameters values was conducted [scikit-learn API]. For each model, dozens of parameter combinations were tested in order to find the most appropriate fit for our data :
For (ii), (vi) an (vii), all combinations of these parameters were tested ('*loss*': ['epsilon insensitive','square epsilon insensitive'], '*C*': [0.001, 0.01, 0.1, 1, 10, 100, 1000]); *loss* determines the type of loss function used to minimize the error. L1 loss function (or epsilon insensitive) is used to minimize the error which is the sum of all of the absolute differences between the true value and the predicted value. L2 Loss Function (or squared epsilon insensitive) is used to minimize the error which is the sum of all of the squared differences between the true value and the predicted value. *C* is the penalty parameter of the error term, the bigger this parameter the less regularization is used. For (ii), ('*epsilon*': [0, 0.001, 0.01, 0.1, 1]) was also tested. *epsilon* specifies the range within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value.
For (iii) and (viii), the regularization parameter *alpha* was set to search over 100 points evenly spaced in log space, with a minimum of 10^6 and a maximum of $1.5 * 10^5$. *alpha* specifies the strength of the regularization which improves the conditioning of the problem and reduces the variance of the estimates. Larger values specify stronger regularization.
For (iv), (v), (ix), (x) and (xi), all combinations of these parameters were tested ('*n estimators*': [50, 100, 150, 200], '*max depth*': [2, 4, 6, 8, 10], '*min samples split*': [2, 4], '*min samples leaf*': [1, 2], '*max features*': ['auto', 'sqrt', 'log2']); *n estimators* specifies the number of boosting stages or the number of trees in the 'forest', a large number usually results in better performance. *max depth* limits the number of nodes in the tree. The best value depends on the interactions of the input variables. *min samples split* specifies the minimum number of samples required to split an internal node. *min samples leaf* specifies the minimum number of samples required to be at a leaf node. And *max features* specifies the number of features to consider when looking for the best split.

2.2.2 Azimuth

`https://crispr.ml/` `https://github.com/microsoftResearch//azimuth`
Azimuth is the current state of the art regarding ML on-target prediction for mammal sgRNAs [20]. It can detect and score potential sgRNAs in a given gene or sequence, and it can score any given 30mer sgRNA with a NGG" PAM. The score returned ranges between 0 and 1, with 1 being the best on-target efficiency. Azimuth predictive model is a Gradient Boosted Regression Tree (GBRT), which was trained on sgRNA used for mammal CRISPR screens.

2.2.3 CRISPR Efficiency Predictor

`https://www.flyrnai.org/evaluateCrispr/` CRISPR Efficiency Predictor is the current algorithm used in my thesis lab to predict sgRNAs efficiency in *D.mel*. The tool was developed by Housden et al. [8] and reports a predicted efficiency score based on the 20 nucleotides sequence of sgRNAs. The score reflects cumulative p-value for high efficiency based on in vitro data generated using a *Drosophila* cell line, with scores above 7.5 indicating high efficiency gRNAs.

Chapter 3

Results

3.1 Identification of sgRNAs targeting essential genes

Based on a genome-wide pooled CRISPR screen conducted in my thesis lab [9], I worked with the read counts of two replicates, and their average, of 79283 sgRNAs targeting 13645 genes. This screen covers more than 90% of *D.mel* genes. In theory, analyzing sgRNAs targeting essential genes is a good way to accurately assess the efficiency of these sgRNAs since effective sgRNAs will eventually kill the cell, resulting in a lower log2 fold-change compared to the reference. In contrast, ineffective sgRNAs will not knock-out their targeted essential gene and will not kill the cell, so the log2 fold-change will remain close to 0. BAGEL and MAGeCK were used to identify the essential genes from this screen. For MAGeCK the normalized read counts is given as input and a z-score for each 13645 gene is returned. The lower the score the more the gene is expected to be essential. For BAGEL, after read counts normalization, the log2FC for each sgRNA was computed and given as input alongside one of the three different essential/non essential genes lists tested ("RNAi", "bj" or "Hart"). A Bayesian Factor (BF) for each 13645 gene is returned. The higher the score the more the gene is expected to be essential.

Essential genes are supposed to have high expression levels since they are necessary to the cell's survival. This result is illustrated by Viswanatha et. al. [9] in [Figure 3.1] where gene predicted to be essential (low MAGeCK z-score) segregate towards highly expressed genes (log(RPKM+0.01) > 1).

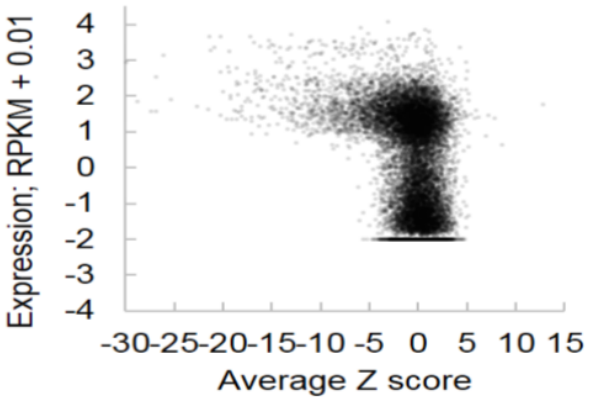


FIGURE 3.1: Expression level (RPKM) of MAGeCK-analyzed genes

Once the 3 replicates (Rep1, Rep2, Avg) have been analyzed with MAGeCK and the different BAGEL schemes ("RNAi", "bj" or "Hart"), a False Discovery Rate analysis of highly expressed genes is conducted in order to select with confidence the most appropriate subset of essential genes and extract the sgRNAs that target them. See Material and Methods for details on MAGeCK, BAGEL and the FDR construction.
In [Figure 3.2], I demonstrate that at 5% False Discovery Rate, analyzing the second replicate, REP2, with MAGeCK leads to the discovery of the largest amount of highly expressed essential genes among all the other replicates and essential genes identification methods. Thus, 1340 highly expressed genes that are predicted to be essential were selected to analyze the efficiency of the 7514 sgRNAs targeting them.

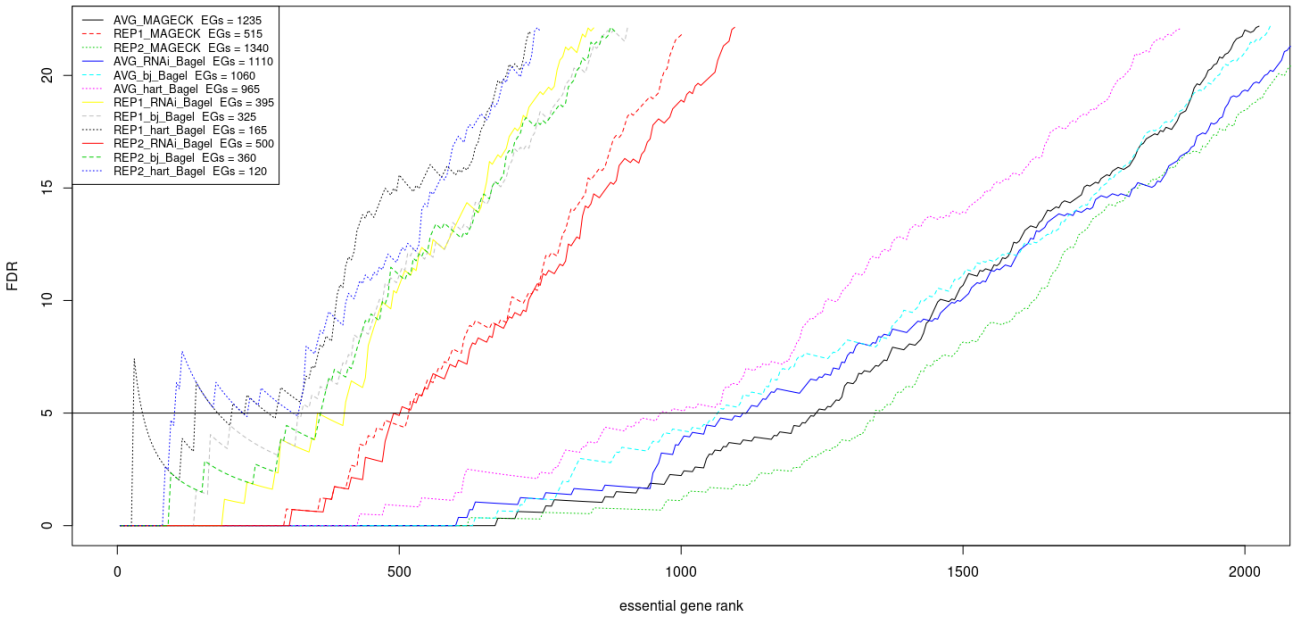


FIGURE 3.2: False Discovery Rate of highly expressed genes according to gene essentiality in CRISPR screen

3.2 Identification of the best ML models to predict *D.mel* sgRNA efficiency

In order to build a tool able to predict the efficiency of a given sgRNA, we first need to identify, for each supervised method (regression and classification), which machine learning algorithms, and their parameters, are the most suited to make such predictions. Since we want to be able to predict the efficiency of different structures of sgRNA (20mer, 23mer or 30mer sgRNA), we are looking for the best machine learning model for each of these 3 structures. Thus, we identify 6 theoretically optimal models, 2 (regression and classification) for each of the 3 sgRNA structures of interest (20, 23 or 30mer).
As the method to identify the best model is the same regardless of the sgRNA structure considered, for regression we will focus on identifying this optimal model for the 20mer sgRNA structure. For classification we will focus on identifying this optimal model for the 23mer sgRNA structure. For each supervised method, the results for the 2 other sgRNA structures are available in appendix. We evaluate the models performance both on the training set of the 7514 sgRNAs targeting essential genes, as well as on a test set of 1030 sgRNAs randomly selected.

3.2.1 Comparison of regression models for 20mer sgRNAs

The efficiency of the 7514 sgRNAs of interest is measured by their log2FC score normalized between 0 and 1, the higher the more efficient is the sgRNA. The goal of regression models is to fit a regression line as best as possible to the distribution of sgRNA efficiency based on their characteristics. The 409 sgRNA features were extracted from their raw sequence thanks to a script written in R, see Material and Methods for featurization details and see section 3.3.2 for script details.
The first crucial step to reduce overfitting and simplify the regression models is to eliminate irrelevant

or redundant features before actually training the model on the most relevant features. To do so a Cross-Validated Recursive Feature Elimination (RFECV) is conducted, see Material and Methods for details. According to [Figure 3.3], by recursively eliminating the worst feature, the coefficient of determination R2 is maximized at around 13.5% for a L2 penalized linear regression model trained with 326 features. The RFECV plots for regression models of 23 and 30mer sgRNAs are available in annex A.2 and A.3.

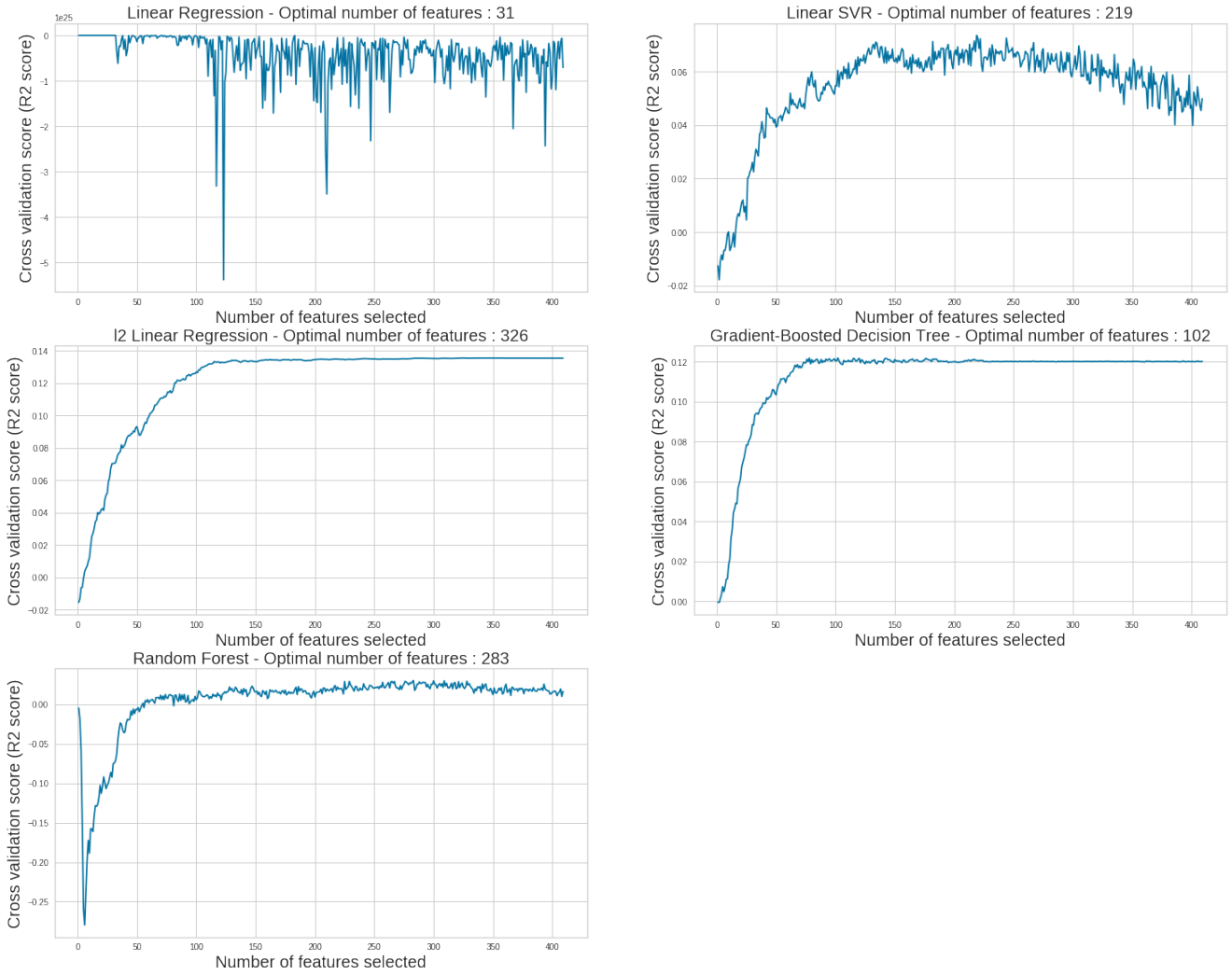


FIGURE 3.3: RFECV to optimize the number of features for regression models of 20mer sgRNAs

Once the optimal numbers of features are found with each algorithm, I used a cross-validated grid search (GridSearchCV) to identify the best hyperparameters for each of the 5 algorithms when trained with each of the 5 potentially optimal numbers of features found by RFECV. So in practice, the best hyperparameters were searched for in 25 different models (5 algorithms * 5 subsets of features). (No figure available, see the Jupyter notebooks [GitHub]).
Now that I have identified the best combination of hyperparameters, I use 10-fold Cross-Validation (CV) to assess the fitting performance of the models [Figure 3.4].
The adjusted R2 is maximized at around 15.1% for a L2-penalized Linear Regression (Ridge) regression model, with the best hyperparameters and trained with 102 features that were selected by Gradient-Boosted Regression Tree (GBRT) RFECV. The other feature selection schemes and algorithms have a

lower adjusted R2 so they are less able to fit the data. The RMSE is about the same at [0.26-0.27] regardless of the model or the feature selection scheme. But the Ridge model with 102 features selected still minimize the RMSE at 0.263

Hence, a Ridge model using 102 features is the best regression model to predict 20mer sgRNA efficiency. A L2-penalized linear regression using 135 features is the best regression model to predict 23mer sgRNA efficiency (see appendix A.4). A linear support vector regression using 121 features is the best regression model to predict 30mer sgRNA efficiency (see appendix A.5).

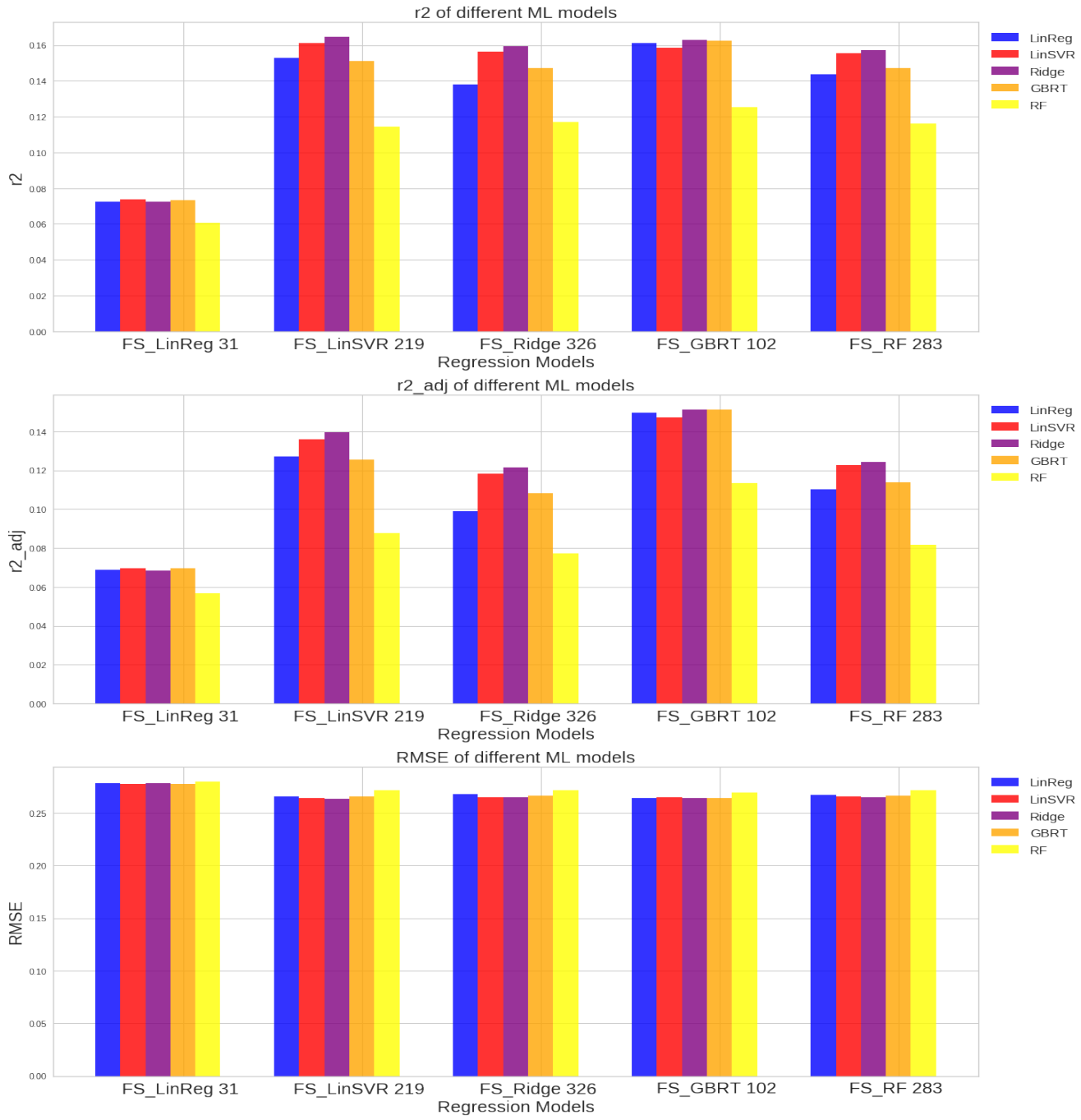


FIGURE 3.4: Performance comparison of the regression models for 20mer sgRNA in each potentially optimal subsets of features

3.2.2 Comparison of classification models for 23mer sgRNAs

sgRNAs are separated in two classes : efficient and non-efficient sgRNAs. sgRNAs with the top 20% Log2FC are considered efficient, and those with the bottom 20% Log2FC are considered non-efficient. The goal of classification models is to find a clear separation between the efficiency classes according to the sgRNAs characteristics.
The protocol to identify the best algorithm to model 23mer sgRNA efficiency is the same as for regression except that the only metric to assess a classification model performance is the area under the receiving operating characteristic (ROC AUC) curve. Hence, for the RFECV feature selection step, we try to maximize the ROC AUC score to find the most appropriate amount of features for each algorithm modelization. According to [Figure 3.5], by recursively eliminating the worst feature, the ROC AUC is maximized at around 77% for a Gradient Boosting Classifier model trained with 125 features. The RFECV plots for regression models of 20 and 30mer sgRNAs are available in appendix A.6 and A.7.

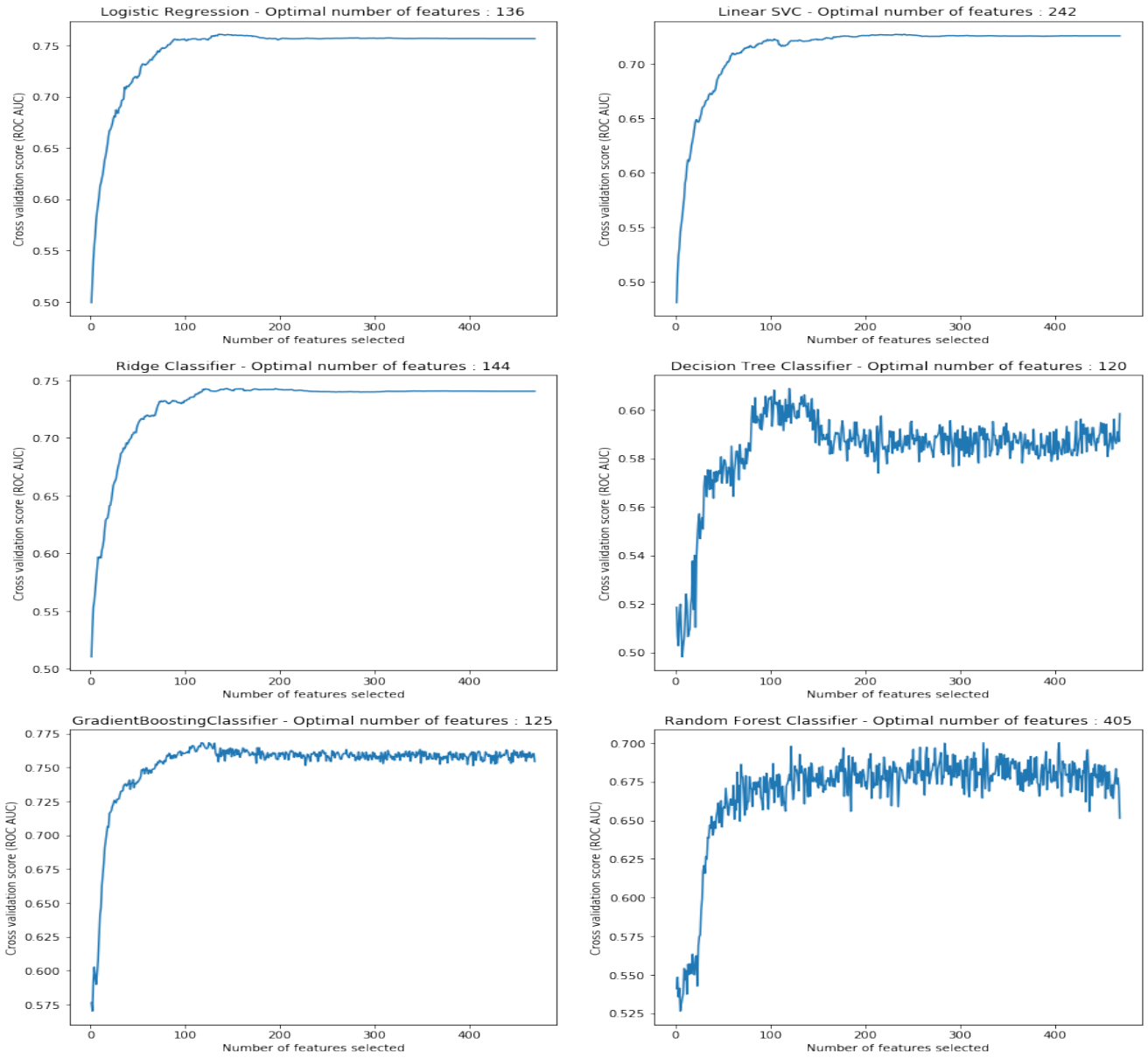


FIGURE 3.5: RFECV to optimize the number of features for classification models of 23mer sgRNAs

As for regression models, (GridSearchCV) was used to identify the best hyperparameters for each of the 6 models when trained with each of the 6 potentially optimal numbers of features found by RFECV. So in practice, the best hyperparameters were searched for in 36 different models (6 algorithms * 6 subsets of features). (No figure available, see the Jupyter notebooks [GitHub]).
Once I have identified the best combination of hyperparameters, I use 10-fold Cross-Validation (CV) to assess the fitting performance of the models [Figure 3.6].
The ROC AUC is maximized at 99% for the Gradient Boosting Classifier (GBC) model, with the best hyperparameters and trained with 405 features (selected by Random Forest Classifier RFECV) or with 120 features (selected by Decision Tree Classifier RFECV). In order to reduce overfitting, the simpler the model the better, so I choose the GBC model with 120 features as the best classification model to predict 23mer sgRNA efficiency. A GBC using 137 features is the best classification model to predict 20mer sgRNA efficiency (see appendix A.8). And a GBC using 400 features is the best classification model to predict 30mer sgRNA efficiency (see appendix A.9).

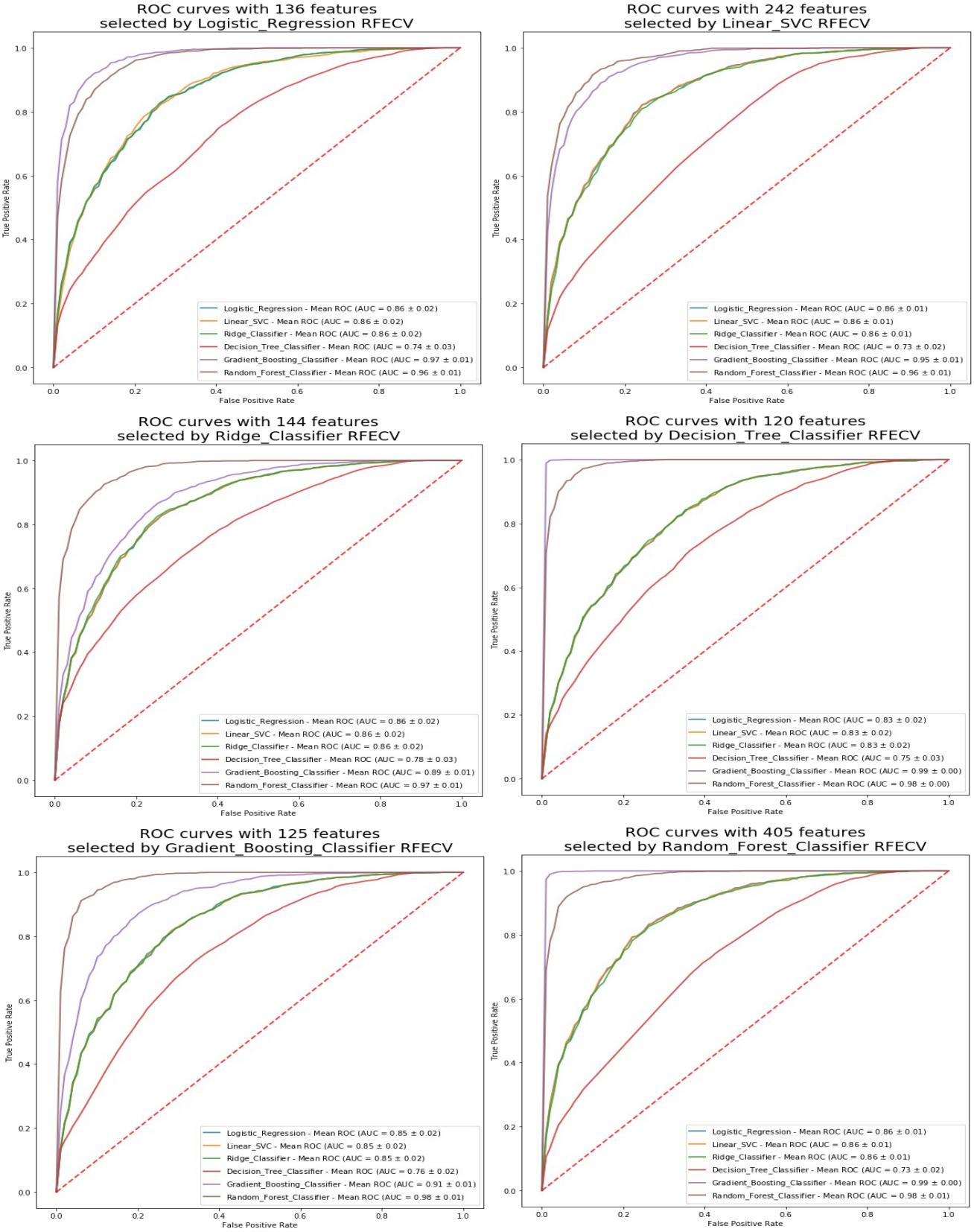


FIGURE 3.6: Performance comparison of the classification models for 23mer sgRNA in each potentially optimal subsets of features

3.2.3 Prediction performance of the selected ML models

Two sets of data are used to assess the accuracy of the sgRNA efficiency predicted my the best models identified. The training set consists of the 7514 sgRNAs used to build the model, and the test sets consist of 1030 sgRNAs chosen from the *D.mel* genome-wide screen (see Material and Methods).
For continuous efficiency (regression models) on the training set, to assess if the predictions are accurate I compare the quartiles of the Log2FC "true efficiency" of the 7514 sgRNAs, to the percentiles of the efficiency predictions from my program [Figure 3.7]. A majority of the sgRNAs predicted to have a high efficiency actually have a high "true" efficiency. I.E among the 316 sgRNAs predicted to have an efficiency of 0.7 to 0.8, around 75% of them have a "true" efficiency of 0.75 to 1.0. The same accuracy pattern is observed for predictions on 23 and 30mer sgRNAs, see Figures A.10 and A.11 in appendix.
For the test set, a similar pattern of efficiency predictions improving proportionally with the "true" efficiency is observed. However, the predictions on the test set correlates more with intermediary "true" efficiencies than predictions on the training set, for several reasons (see Discussion).

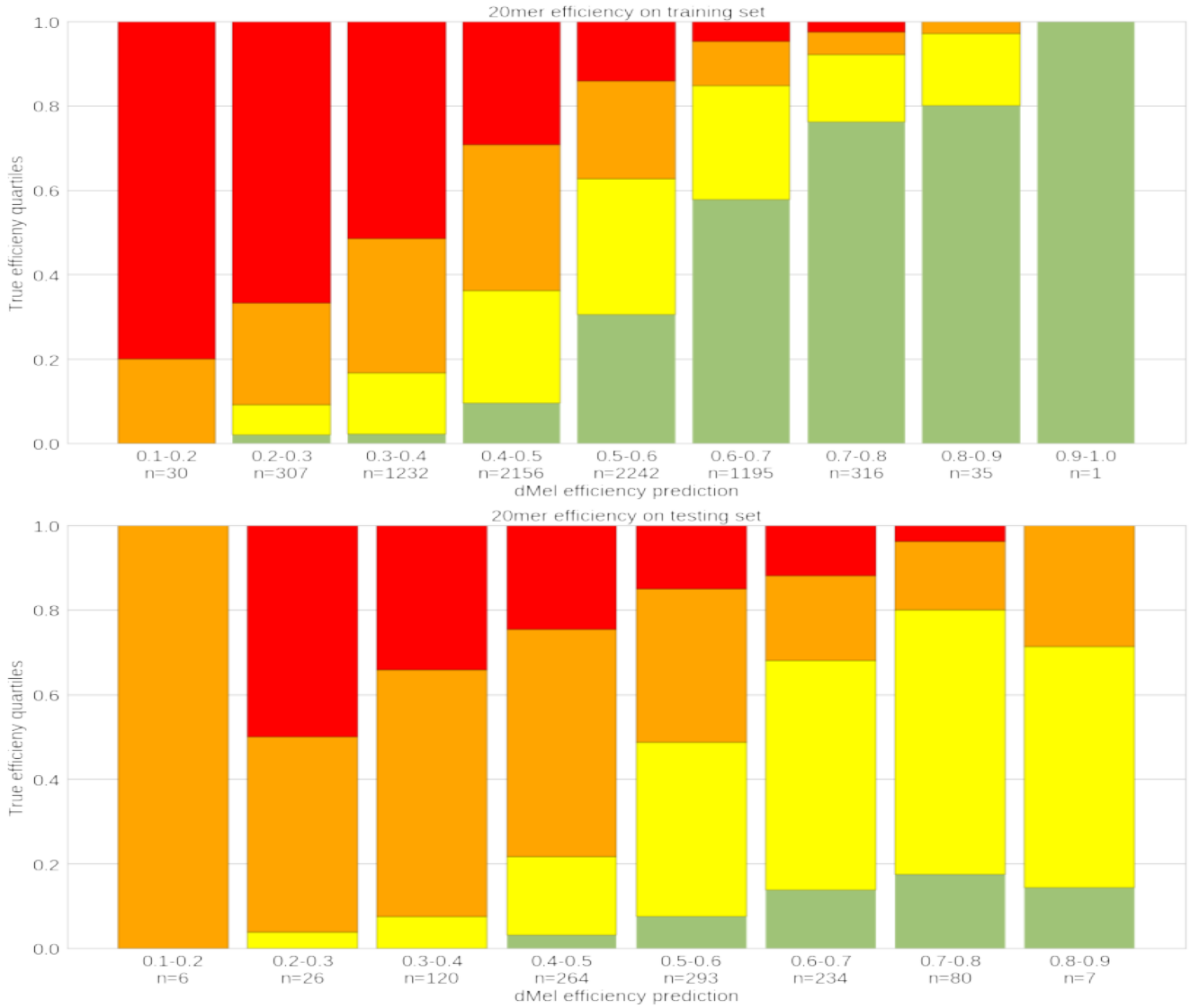


FIGURE 3.7: Performance analysis of the regression model predictions for 20mer sgRNAs. red = 0.0 - 0.25 efficiency, orange = 0.25 - 0.50 efficiency, yellow = 0.50 - 0.75 efficiency, green = 0.75 - 1.0 efficiency.

For binary efficiency (classification models) on the training set, to assess if the predictions are accurate I compare the percentiles of the Log2FC "true efficiency" of the 7514 sgRNAs, to the binary efficiency predictions from my program [Figure 3.9]. A majority of the sgRNAs predicted to have a low efficiency actually have a low "true" efficiency. I.E. among the 759 sgRNAs that have an actual efficiency of 0.2 to 0.3, around 30% of them are predicted to be efficient sgRNAs. The same accuracy pattern is observed for predictions on 23 and 30mer sgRNAs, see Figures A.12 and A.13 in appendix. For the test set, I assume that the top and bottom 20% sgRNAs of the dataset are respectively efficient and non efficient (about 200 sgRNAs in each class), and I compare the performance of the classifications with a ROC curve [Figure 3.8]. The models make accurate classification predictions with an AUC of around 80%. When comparing the classification predictions to the continuous efficiency score, a clearer split at 0.5 efficiency is observed. However, the model appears biased towards classifying sgRNAs in the "efficient" class. Indeed around 40%-50% of sgRNAs that should be classified as "non-efficient" are still classified as "efficient".

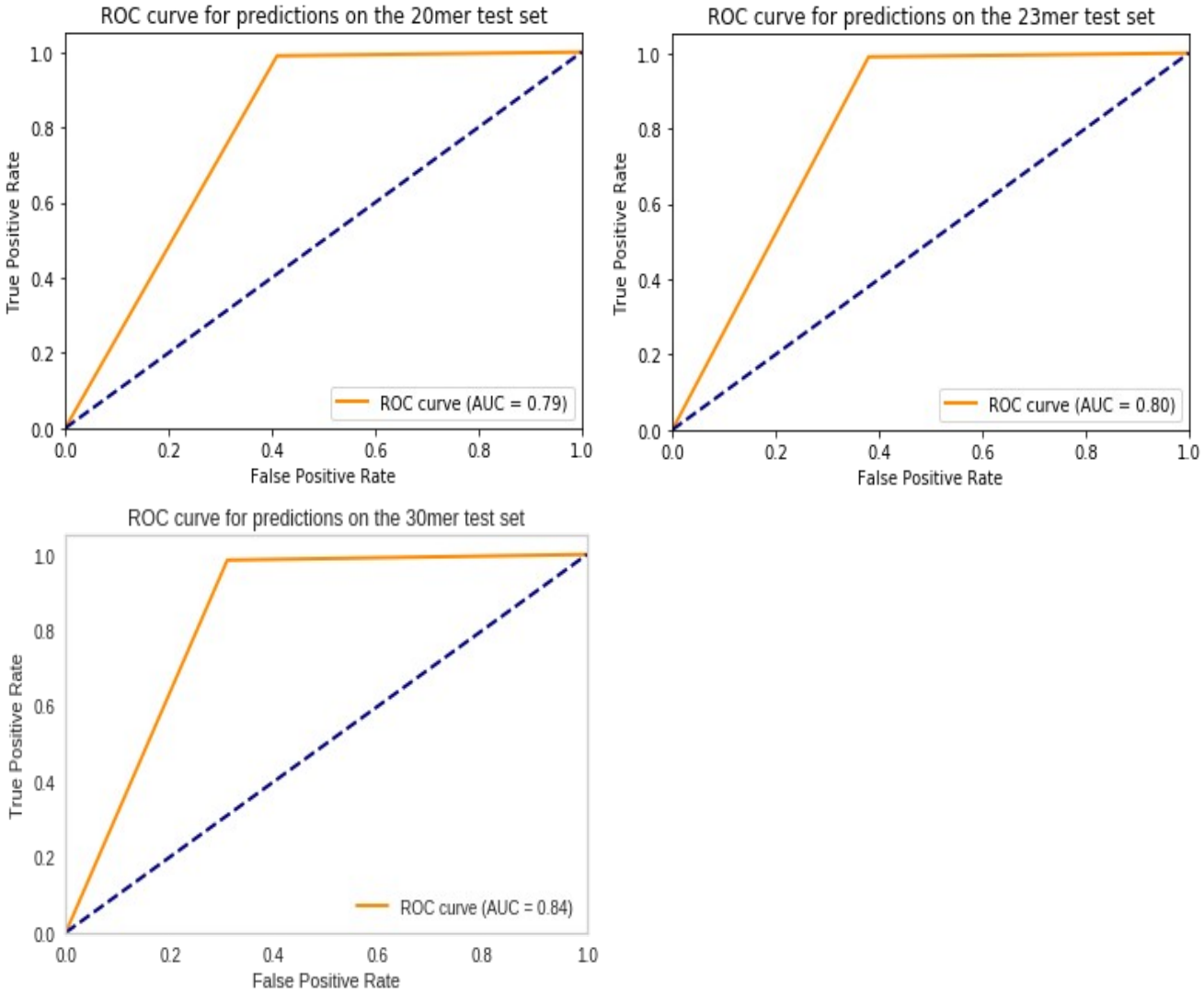


FIGURE 3.8: Classification performance on the test set for the 3 GBC models

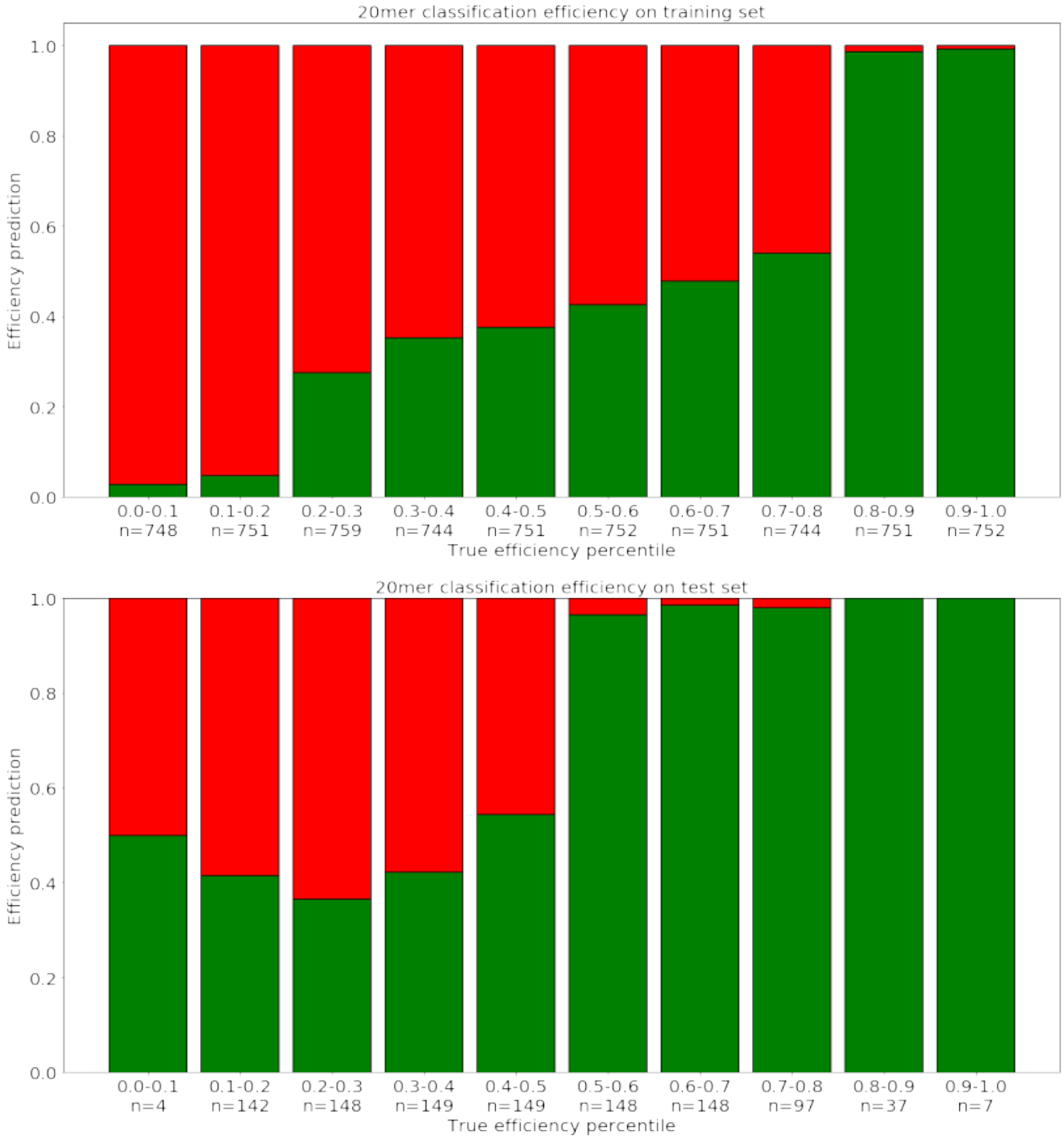


FIGURE 3.9: Performance analysis of the classification model predictions for 20mer sgRNAs. red = 'low predicted efficiency. green = 'high predicted efficiency'.

3.3 Development of dMel sgRNA Efficiency Prediction tool

https://github.com/PierreMkt/Dmel-sgRNA-Efficiency-Prediction

Now that the best regression and classification models have been identified for each of the 3 sgRNA structures of interest (20, 23 and 30mer), the goal is to build a tool able to make efficiency predictions while only having to input the raw sequence of any sgRNA. See Appendix B for the python code.
To do so, the python *Pickle* library is used to save the 6 ML models and the relevant features associated.

Thus, the models can easily be "unpickled" (loaded) to make predictions.

3.3.1 API

The program requires Python3, R and the following python libraries : pandas, numpy, scikit-learn and scipy. After cloning or downloading the GitHub repository, to install the required python packages simply run in the terminal :
pip3 install -r /PATH_TO/utils/requirements.txt
The required R packages, such as *stringr*, will be installed automatically upon running the program.
4 arguments can be given as inputs :

1. Either :
   - - - *seq* : One raw 20, 23 or 30mer sgRNA sequence.
   - - - *csv* : A *.csv* file with a header and the 20, 23 or 30mer sgRNAs listed beneath
2. - - *out* : (optional argument) Path and/or name of the predictions output *.csv* file.
   Default is PATH_TO_FOLDER/sgRNA_efficiency_prediciton/*N*mer_sgRNA_predictions.csv
3. - - *bin* : (optional argument) Whether or not to include binary classification predictions of the sgRNA efficiency in the output. Default is 'no'.

For example, to predict the efficiency of a single 23mer sgRNA and print the classification prediction, type in the terminal:
python3 dMel_CRIPSR_efficiency.py –seq GGAGGCTGCTTTACCCGCTGTGG –bin yes

The output will print the efficiency predictions from the regression and classification models, as well as the path were the *.csv* file results was created. I.E. :
Results exported to /home/.../sgRNA_efficiency_prediction/23mer_sgRNA_predictions.csv
Predicted efficiency score [0:1] for GGAGGCTGCTTTACCCGCTGTGG = [0.50718358] (the higher the better).
The classification model predicts that GGAGGCTGCTTTACCCGCTGTGG will be an effective sgRNA.

To predict the efficiency of many 30mer sgRNAs listed in a *.csv* file, type in the terminal:
python3 dMel_CRIPSR_efficiency.py –csv TEST_sgRNA30mer_to_predict.csv –bin yes

The output will print the path were the *.csv* file results was created. I.E. :
Results exported to /home/.../sgRNA_efficiency_prediction/30mer_sgRNA_predictions.csv

3.3.2 In-depth description

sgRNA Feature Extraction. After parsing the arguments and checking that the raw sequence format is right, or that the file inputed is a *.csv*, the program launches a R script that extracts the features from the raw sgRNA sequences. First, each sgRNA characteristic is positioned in a specific order depending on the structure of the sgRNA inputed (20, 23 or 30mer). Indeed, the position of each feature must exactly match the position in which were the features used to build the ML models. For a 30mer sgRNA, the order of the features position I arbitrarily chose is :
GC content of the 20mer sgRNA ; Melting temperatures for the start [1-7], middle [8-15] and end [16-20] of the 20mer ; Melting temperature of the full 20, 23 or 30mer sgRNA ; position independent order 1 (4 features) ; position independent order 2 (16 features) ; position dependent order 1 (120 features);

position dependent order 2 (464 features) ; 16 possible combinations of dinucleotide flanking the PAM. 20 and 23mer sgRNAs have less position dependent features since their sequence is shorter, and they do not have the PAM feature since the 24th nucleotide is unknown.
Once the sgRNA characteristics are set, the program goes through the sgRNA inputed, or each sgRNA of the *.csv* file, it "extracts" the features by calculating the GC%, the melting temperatures and scanning the sgRNA to retrieve the amount and positions of each (di)nucleotide, and it then populates the table of all the features. This table is saved under *R_Featurized_sgRNA.csv*.

Feature selection and efficiency predictions. The continuous features of the table are standard scaled. Then the appropriate ML model, and the index of the features it uses, are loaded depending on the structure of the sgRNA(s) inputed. Using the index of relevant features, the irrelevant features are eliminated and then the model predicts the efficiency of the sgRNA(s) by analyzing the values of each feature of interest from the table.

Outputs. The R script outputs the table of featurized sgRNAs. The python script outputs a *.csv* file of the sgRNA(s) prediction(s). If the - - *seq* argument is used, the prediction results are also outputted directly in the terminal.

3.4 dMel sgRNA Efficiency Prediction tool has better performance than CRISPR Efficiency Predictor and Azimuth

Now that the prediction tool is complete, I assess its predictions accuracy and compare its performance to CRISPR Efficiency Predictor and Azimuth.

3.4.1 Comparison of predictions performance between my program, Azimuth and CRISPR Efficiency Predictor with *Drosophila* and *Human* datasets

To compare the prediction performances of these different tools I use the 2 test sets (none of the model used as been build using this dataset) and plot the predicted efficiency against the actual "true" efficiency, draw the regression line and calculate the adjusted R2 as well as the RMSE, with *Drosophila* or *Human* datasets.

First I compare the predictions with the *D.mel* test set [Figure 3.10]. CRISPR Efficiency Predictor, the tool currently used in my thesis lab, shows poor performance with an adjusted R2 and a linear regression slope almost null, as well as a higher RMSE than with the other predictions tools. Indeed, Azimuth and all the prediction models from my tool show good prediction performance, meaning that the efficiency predictions are close the actual efficiency of the guides. My program has higher linear regression slope, higher adjusted R2 and lower RMSE than Azimuth. Thus, for *D.mel* sgRNAs, my program has better performance than Azimuth, the current state of the art for sgRNA efficiency prediction.

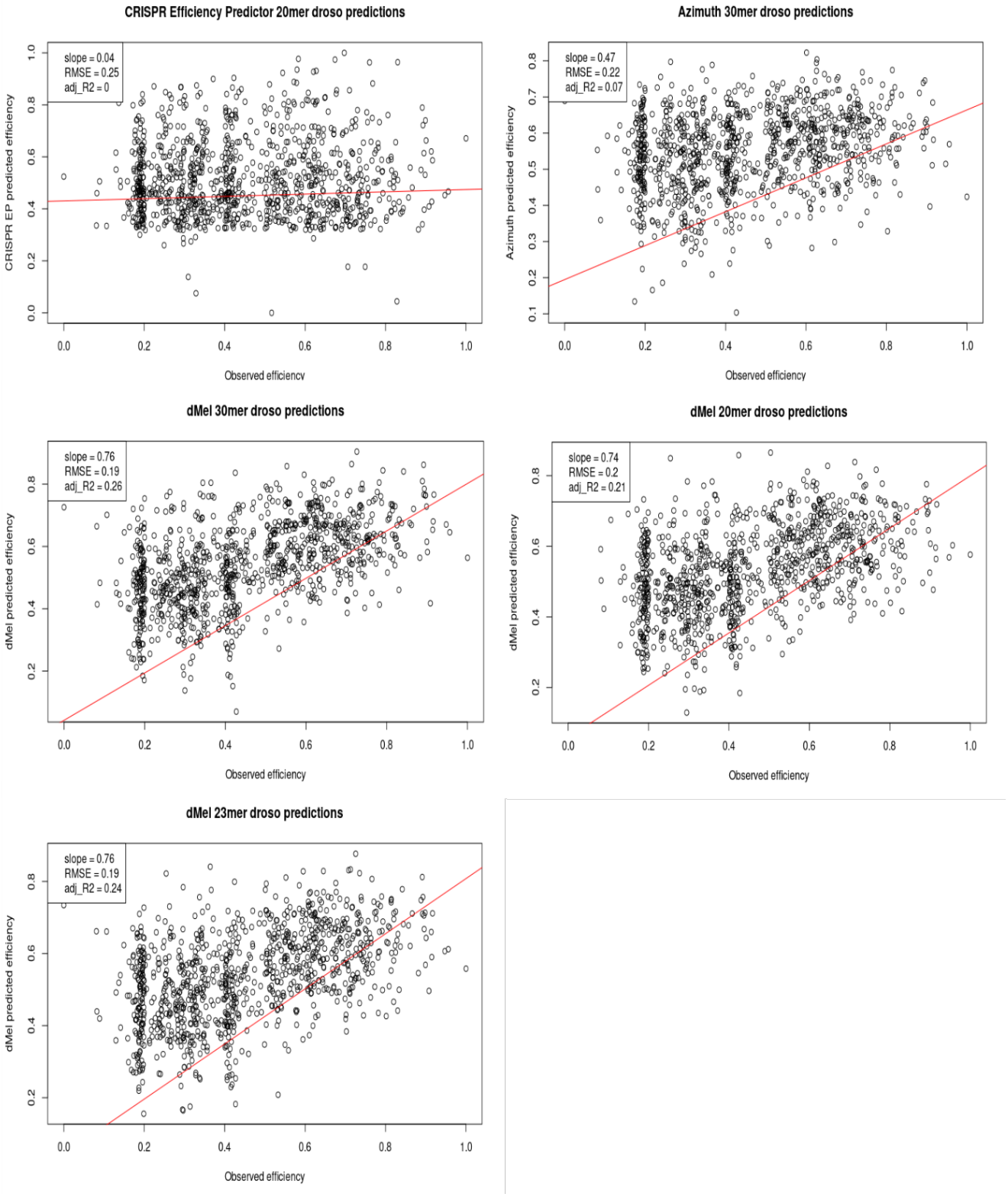


FIGURE 3.10: Performance comparison of efficiency prediction tools on the *Drosophila* testing set.

Then I compare the predictions of Azimuth and my program with the *Human* test set [Figure 3.11]. The 3 performance metrics considered (linear regression slope, RMSE and adjusted R2) are more optimized with Azimuth predictions than with predictions from my program. Hence, for *Human* sgRNAs,

Azimuth is more accurate at predicted sgRNA efficiency than my program.

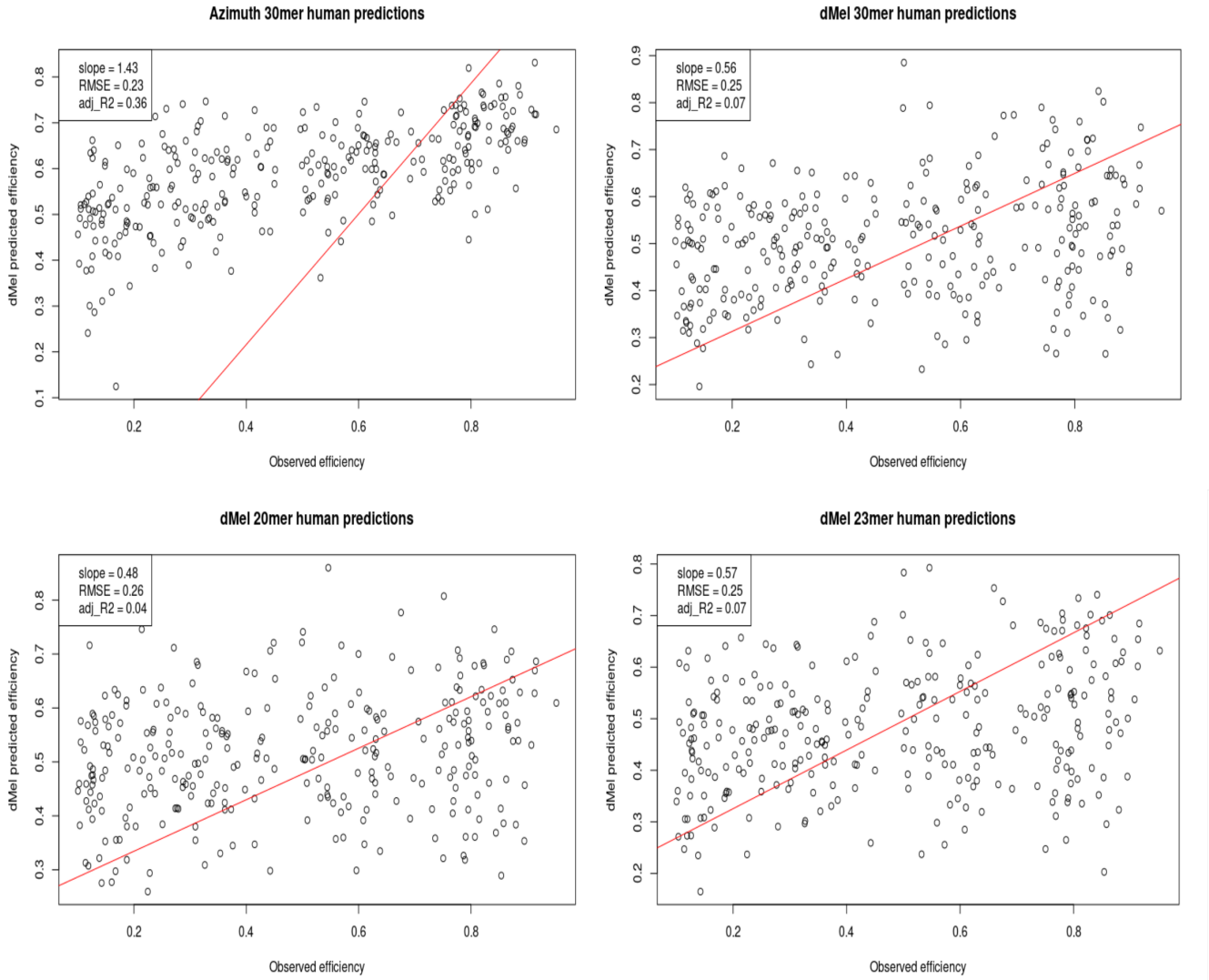


FIGURE 3.11: Performance comparison of efficiency prediction tools for 2600 *Human* sgRNAs.

Chapter 4

Discussion

4.1 The limitations of sgRNA efficiency prediction

4.1.1 Suboptimal dataset used to build the predictive model

The genome-wide CRISPR screen has 79283 sgRNAs targeting 13645, resulting in around 6 sgRNAs per gene. After the essential genes identification phase, we retrieve 7514 essential genes. Analyzing a lot of genes, each targeted by a few sgRNAs, may be suboptimal to accurately identify sgRNA efficiency. Indeed, all genes are not equally essential, nor are they expressed at the same levels. Therefore, following Doench et.al. protocol [20], I suggest that analyzing a few genes targeted by a lot of sgRNAs may be a better approach to identify sgRNA efficiency. To do so, genes with a large sequence may be preferred in order to be able to design a lot of sgRNAs all along the sequence and on both strands.
Moreover, using essential genes to identify sgRNA efficiency may also be suboptimal. Indeed, a "bad" sgRNA that has poor on-target efficiency but lots of off-targets will cause many mutations, potentially kill the cell and be falsely interpreted as a "good" sgRNA even though it has not edited its expected target. Therefore, following Doench et.al. protocol, I suggest that targeting easily identifiable, but non essential, genes such as cell surface markers, may be a better approach to accurately identify sgRNA efficiency.

4.1.2 Unconsidered features to promote generalization

I chose to only build the ML models based on the sgRNA sequence in order to promote generalization and ease of use of the subsequent efficiency prediction program I developed. Indeed, many other relevant features based on the targeted sequence would have been relevant for the ML models, such as the chromatin state, the expected location of the mutation in the mRNA or the distance between the start of the targeted exon and the PAM sequence. However for my program, in order to properly retrieve the values of these features the user would have had to provide the characteristics of the targeted DNA region alongside the sgRNA. But not all user would have this information when willing to design sgRNAs. Hence my choice to only focus on features that can be extracted from the raw sgRNA sequence.

4.2 Machine Learning models analysis

4.2.1 Description of the best algorithms selected

For regression, the best ML algorithms identified are L2-penalized Linear Regression (Ridge) and Linear Support Vector Regression (LinSVR). Briefly, Ridge adds a regularization term (the sum of the square of the weights) to the regression line in order to prevent the coefficients to fit perfectly and overfit. LinSVR

searches for the regression line that is at most *epsilon* distance from the "true" efficiency. Linear SVR have similar performance as regular SVR but trains much faster.

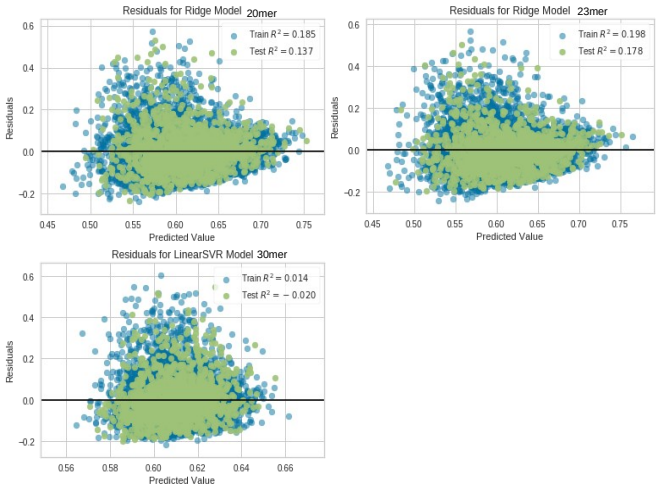


FIGURE 4.1: Residuals plots of the best selected models

Even though these algorithms were the best among the ones I chose to analyze, they may not be the most appropriate to model the data. Indeed, the coefficient of determination is still very low (15%-20%), meaning that only a fraction of the selected features actually explain the variation in sgRNA efficiency. Nevertheless, some field of study are harder to model and lower R2 are expected (e.g. any field that attempts to predict human behavior, such as psychology, typically has R2 values lower than 50%). Furthermore, R2 does not assess if the predictions are biased, which is why it is important to assess the plot of residuals that maps the difference between the "true" sgRNA efficiency and the predicted efficiency. In the linear regression context, random errors are assumed to produce residuals that are normally distributed. Therefore, the residuals should fall in a symmetrical pattern and have a constant spread throughout the range. However in [Figure 4.1] the residuals do no seem randomly distributed, meaning that there are still patterns in the data that the models do not manage to explain, and that linear regression algorithms are probably not the most appropriate algorithms to model the data. More complex algorithms could be used such as non-linear regression, SVR or tree-based models. However, I did not manage to efficiently implement non-linear regression in python, SVR with a radial basis function kernel (rbf) is computationally heavy and showed worse performance than Linear SVR (data not shown), and tree-based models were used but still present worse performance than linear regression models (Ridge, Linear SVR). Other extremely powerful and popular methods available are Convolutional Neural Networks (CNN) and Deep Learning [27] but I did not have the time to train myself in these techniques. So, I sticked with the linear models I identified.

For classification, the best ML algorithm identified for the 3 sgRNA structures is Gradient-Boosting Classifier (GBC). Briefly, Gradient boosting involves three elements: a loss function to be optimized, a tree-based "weak learner" to make predictions and an additive model to add weak learners to minimize the loss function. The output for the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve the final output of the model. Tree addition stops once a fixed number of trees are added or once the training error reaches an acceptable level or no longer improves. https://bit.ly/2Gb6Sp2
The GBC performance are nearly perfect (ROC AUC = 99%) which concerned me because this is a sign of overfitting. If the model was overfitted, when its performance were evaluated on the test set we would have observed very poor classifications. However, despite the small bias towards the efficient class, I am satisfied with the performance of the models (AUC = 80% on the test set).

4.2.2 Relevant features

One of the key goals of sgRNA design optimization is to discover universal rules regarding the structure of efficient and non-efficient sgRNAs. Instinctively, we may think that retrieving the coefficients of the

ML model would provide the importance of each feature individually, which is the case for simpler algorithms such as Linear Regression.

However, more complex algorithms take into consideration the interactions between each feature before assigning the weights. Thus, a negative coefficient for a feature does not necessarily mean that sgRNAs with this feature have lower efficiency than sgRNAs without the feature. Therefore, it is more appropriate to analyze the *importance* of the features, or in other words, how much a feature contributes to the accuracy of the ML model, may it be positively or negatively. We can retrieve such information with a GBRT model [Figure 4.2]. The One-hot-encoded possible positions of each possible dinucleotide (dependent order 2) are the features providing the most information to the model. But taken individually, most of these features have an importance of less than 1%.

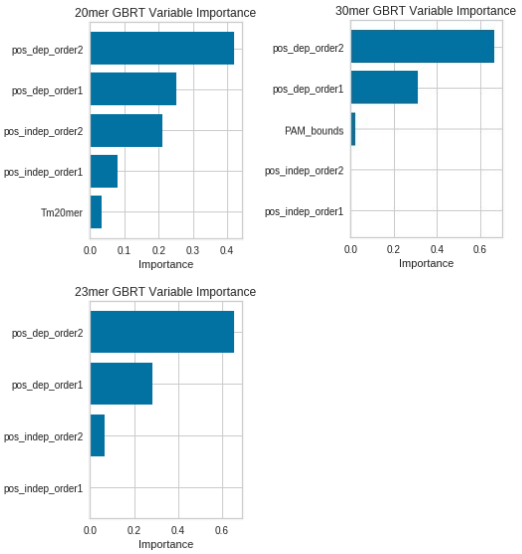


FIGURE 4.2: Feature Importance with GBRT models

4.2.3 Performance of dMel sgRNA Efficiency Prediction tool

The sgRNA efficiency predictions are more accurate on the training set than on the test set, and this is to be expected. Indeed, since the ML model has been trained on the training set we expect its performance to be very accurate but not perfect as it is crucial to not overfit the model and preserve generalization. Therefore, the fact that the predictions on the test set show similar patterns of performance as predictions on the training set is very promising. The ML models developed are generalizable and accurate enough to make precise predictions of sgRNA efficiency.
When comparing the metrics used to assess prediction accuracy, my program is more accurate than Azimuth and CRISPR Efficiency Predictor (CEP) when predicting *D.mel* sgRNAs efficiency. And its performance when predicting *Human* sgRNAs is a little worse than Azimuth, the state of the art in sgRNA efficiency prediction. Thus, dMel sgRNA Efficiency Prediction should be powerful tool to help my thesis lab, the DRSC and the overall *Drosophila* scientific community design more efficient sgRNAs.

4.3 Conclusion and Future Directions

In this project my aim was to provide the Drosophila RNAi Screening Center (DRSC) with an accurate CRISPR sgRNA efficiency scoring tool in order to improve the sgRNA design pipeline. To do so I used ML algorithms to model the efficiency of sgRNAs targeting essential genes, based on their characteristics, in a *D.mel* genome-wide pooled CRISPR screen. The ML models identified were implemented in a program with which single or batch sgRNA efficiency predictions can be made. Finally, for sgRNAs designed for the *D.mel* genome, my program showed better performance than CEP, the tool currently used in the DRSC, and Azimuth, the current sgRNA efficiency prediction state-of-the-art. Hence, I advise the *Drosophila* community to use my tool to assess the theoretical efficiency of their sgRNAs.
The next step in this project would be to implement my tool in the DRSC Find CRISPRs online tool so that users can more easily curate their choice of sgRNA via my program.
The main area of improvement for this program, and for the sgRNA design improvement field in general,

would be to use more powerful and cutting-edge ML algorithms such as Deep Learning [27] or CNN to model more effectively the sgRNA efficiency according to the sgRNA characteristics. Also, another important area of improvement would be to design pooled CRISPR screens more appropriate to sgRNA efficiency analysis, i.e. not focusing on sgRNAs targeting essential genes, and using a few genes targeted by many sgRNAs.

Bibliography

[1] Ophir Shalem et al. “Genome - scale CRISPR - Cas9 knockout screening in human cells”. In: *Science* 343.6166 (2014), pp. 84–87. ISSN: 1095-9203. DOI: 10.1126/science.1247005.Genome-Scale. arXiv: NIHMS150003.

[2] Yuxuan Wu et al. “Correction of a genetic disease in mouse via use of CRISPR-Cas9”. In: *Cell Stem Cell* 13.6 (2013), pp. 659–662. ISSN: 19345909. DOI: 10.1016/j.stem.2013.10.016. arXiv: 38. URL: http://dx.doi.org/10.1016/j.stem.2013.10.016.

[3] Andrew R. Bassett et al. “Highly Efficient Targeted Mutagenesis of Drosophila with the CRISPR/-Cas9 System”. In: *Cell Reports* 4.1 (2013), pp. 220–228. ISSN: 22111247. DOI: 10.1016/j.celrep.2013.06.020.

[4] Hui Zhang et al. “Genome Editing—Principles and Applications for Functional Genomics Research and Crop Improvement”. In: *Critical Reviews in Plant Sciences* 36.4 (2017), pp. 291–309. ISSN: 15497836. DOI: 10.1080/07352689.2017.1402989. URL: https://doi.org/10.1080/07352689.2017.1402989.

[5] Yujia Caia, Rasmus O. Baka, and Jacob Giehm Mikkelsena. “Targeted genome editing by lentiviral protein transduction of zinc-finger and TAL-effector nucleases”. In: *eLife* 2014.3 (2014), pp. 1–19. ISSN: 2050084X. DOI: 10.7554/eLife.01911. arXiv: /www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3006164{\&}tool=pmcentrez{\&}rendertype=abstract. [Figures, S., 2010. Supplementary information. Nature, 1(c), pp.1-7. Available at: http:].

[6] Stephanie Mohr, Chris Bakal, and Norbert Perrimon. “Genomic Screening with RNAi: Results and Challenges”. In: *Annual Review of Biochemistry* 79.1 (2010), pp. 37–64. ISSN: 0066-4154. DOI: 10.1146/annurev-biochem-060408-092949. arXiv: NIHMS150003. URL: http://www.annualreviews.org/doi/10.1146/annurev-biochem-060408-092949.

[7] Stephanie E. Mohr et al. “RNAi screening comes of age: improved techniques and complementary approaches.” In: *Nature Review Molecular and Cell Biology* 15.9 (2015), pp. 591–600. DOI: 10.1038/nrm3860.RNAi.

[8] Benjamin E. Housden et al. “Identification of potential drug targets for tuberous sclerosis complex by synthetic screens combining CRISPR-based knockouts with RNAi”. In: *Science Signaling* 8.393 (2015), pp. 1–10. ISSN: 19379145. DOI: 10.1126/scisignal.aab3729.

[9] Raghuvir Viswanatha et al. “Pooled genome-wide CRISPR screening for basal and context-specific fitness gene essentiality in Drosophila cells Running Title: Pooled genome-wide CRISPR screens in Drosophila”. In: *bioRxiv* 617 (2018), pp. 1–20. ISSN: 2050-084X. DOI: 10.1101/274464. URL: https://www.biorxiv.org/content/early/2018/03/01/274464%7B%5C%%7D0Ahttp://dx.doi.org/10.1101/274464.

[10] Yanfang Fu et al. “Improving CRISPR-Cas nuclease specificity using truncated guide RNAs”. In: 32.3 (2014), pp. 279–284. DOI: 10.1038/nbt.2808.Improving.

[11] Vineeta Agarwala et al. "DNA targeting specificity of RNA-guided Cas9 nucleases". In: *Nature Biotechnology* 31.9 (2014), pp. 827–832. DOI: 10.1038/nbt.2647.DNA.

[12] Le Cong et al. "Multiplex Genome Engineering Using CRISPR/VCas Systems". In: *Science* 339.6121 (2013), pp. 819–823. ISSN: 15378276. DOI: 10.1126/science.1231143.Multiplex. arXiv: NIHMS150003.

[13] T Wang et al. "Genetic screens in human cells using the CRISPR/Cas9 system". In: *Science* 343.6166 (2014), pp. 80–84. ISSN: 1095-9203. DOI: 10.1126/science.1246981.Genetic. arXiv: NIHMS150003. URL: http://www.sciencemag.org/content/343/6166/80.short.

[14] X. Ren et al. "Optimized gene editing technology for Drosophila melanogaster using germ line-specific Cas9". In: *Proceedings of the National Academy of Sciences* 110.47 (2013), pp. 19012–19017. ISSN: 0027-8424. DOI: 10.1073/pnas.1318481110. arXiv: arXiv:1408.1149. URL: http://www.pnas.org/cgi/doi/10.1073/pnas.1318481110.

[15] Raj Chari et al. "Unraveling CRISPR-Cas9 genome engineering parameters via a library-on-library approach". In: *Nature Methods* 12.9 (2015), pp. 823–826. ISSN: 15487105. DOI: 10.1038/nmeth.3473. arXiv: 15334406. URL: http://dx.doi.org/10.1038/nmeth.3473.

[16] Han Xu et al. "Sequence determinants of improved CRISPR sgRNA design". In: *Genome Research* 25.8 (2015), pp. 1147–1157. ISSN: 15495469. DOI: 10.1101/gr.191452.115.

[17] Traver Hart et al. "Evaluation and Design of Genome-Wide CRISPR/SpCas9 Knockout Screens". In: *G3: Genes|Genomes|Genetics* 7.8 (2017), pp. 2719–2727. ISSN: 2160-1836. DOI: 10.1534/g3.117.041277. URL: http://g3journal.org/lookup/doi/10.1534/g3.117.041277.

[18] Maximilian Haeussler et al. "Evaluation of off-target and on-target scoring algorithms and integration into the guide RNA selection tool CRISPOR". In: *Genome Biology* 17.1 (2016), pp. 1–12. ISSN: 1474760X. DOI: 10.1186/s13059-016-1012-2. URL: http://dx.doi.org/10.1186/s13059-016-1012-2.

[19] John G. Doench et al. "Rational design of highly active sgRNAs for CRISPR-Cas9- mediated gene inactivation". In: *Nature Biotechnology* 32.12 (2014), pp. 1262–1267. ISSN: 00029378. DOI: 10.1038/nbt.3026.Rational. arXiv: NIHMS150003.

[20] John G. Doench et al. "Optimized sgRNA design to maximize activity and minimize off-target effects of CRISPR-Cas9". In: *Nature Biotechnology* 34.2 (2016), pp. 184–191. ISSN: 15461696. DOI: 10.1038/nbt.3437. arXiv: 15334406.

[21] Traver Hart and Jason Moffat. "BAGEL: A computational framework for identifying essential genes from pooled library screens". In: *BMC Bioinformatics* 17.1 (2016), pp. 1–7. ISSN: 14712105. DOI: 10.1186/s12859-016-1015-8. arXiv: arXiv:0907.2398v1. URL: http://dx.doi.org/10.1186/s12859-016-1015-8.

[22] Wei Li et al. "MAGeCK enables robust identification of essential genes from genome-scale CRISPR/-Cas9 knockout screens". In: *Genome biology* 15.12 (2014), p. 554. ISSN: 1474760X. DOI: 10.1186/s13059-014-0554-4. arXiv: arXiv:1011.1669v3.

[23] Yanhui Hu et al. "An integrative approach to ortholog prediction for disease-focused and other functional studies". In: *BMC Bioinformatics* 12 (2011). ISSN: 14712105. DOI: 10.1186/1471-2105-12-357.

[24] Michael Boutros et al. "Genome-Wide RNAi Analysis of Drosophila Cells". In: *Society* 303.February (2004), pp. 832–835. URL: file:///Users/Guy%20F/Documents/Mendeley%20Desktop/Boutros%20et%20al/Society/Boutros%20et%20al.%20-%202004%20-%20Genome-Wide%20RNAi%20Analysis%20of%20Drosophila%20Cells.pdf.

[25] Mikael Björklund et al. "Identification of pathways regulating cell size and cell-cycle progression by RNAi". In: *Nature* 439.7079 (2006), pp. 1009–1013. ISSN: 14764687. DOI: 10.1038/nature04469.

[26] T. Hart et al. "Measuring error rates in genomic perturbation screens: gold standards for human functional genomics". In: *Molecular Systems Biology* 10.7 (2014), pp. 733–733. ISSN: 1744-4292. DOI: 10.15252/msb.20145216. URL: http://msb.embopress.org/cgi/doi/10.15252/msb.20145216.

[27] Hui Kwon Kim et al. "Deep learning improves prediction of CRISPR-Cpf1 guide RNA activity". In: *Nature Biotechnology* 36.3 (2018), pp. 239–241. ISSN: 15461696. DOI: 10.1038/nbt.4061.

List of Abbreviations

ML	**M**achine **L**earning
D.mel	**D**rosophila **M**elanogaster
CRISPR	**C**lustered **R**egularly **I**nterspaced **S**hort **P**alindromic **R**epeats
Cas9	**C**RISPR **a**ssociated (protein) **9**
sgRNA	**s**ingle **g**uide **R**ibo **N**ucleic **A**cid
crRNA	**cr**ispr **RNA**
tracrRNA	**tr**ans-**a**ctivating **crRNA**
PAM	**P**rotospacer **A**djacent **M**otif
DRSC	**D**rosophila **R**NAi **S**creening **C**enter
DSBs	**D**ouble **S**trand **B**reak**s**
HDR	**H**omology **D**irected **R**epair
NHEJ	**N**on **H**omologous **E**nd **J**oining
ZFN	**Z**inc **F**inger **N**uclease
TALEN	**T**ranscription **A**ctivator **L**ike **E**ffector **N**uclease
Log2FC	**Log2 F**old **C**hange
DIOPT	**D**RSC **I**ntegrative **O**rtholog **P**rediction **T**ool
MAGeCK	**M**odel-based **A**nalysis of **Ge**nomic-wide **C**RISPR-Cas9 **K**nockout
BAGEL	**B**ayesian **A**nalysis of **G**ene **E**ssentiality
FPKM	**F**ragments **P**er **K**ilobase of transcript per **M**illion
RPKM	**R**eads **P**er **K**ilobase of transcript per **M**illion
FDR	**F**alse **D**iscovery **R**ate
SVM	**S**upport **V**ector **M**achine
LinSVR	**Lin**ear **S**upport **V**ector **R**egression
GBRT	**G**radient-**B**oosted **R**egression **T**ree
RF	**R**andom **F**orest
RFC	**R**andom **F**orest **C**lassifier
DTC	**D**ecision **T**ree **C**lassifier
GBC	**G**radient-**B**oosting **C**lassifier
RMSE	**R**oot **M**ean **Square E**rror
ROC	**R**eceiving **O**perating **C**haracteristic
AUC	**A**rea **U**nder the **C**urve
CV	**C**ross-**V**alidation
RFECV	**R**ecursive **F**eature **E**limination with **C**ross-**V**alidation
CEP	**C**RISPR **E**fficiency **P**redictor
API	**A**pplication **P**rogram **I**nterface
CNN	**C**onvolutional **N**eural **N**etwork

Appendix A

Appendix

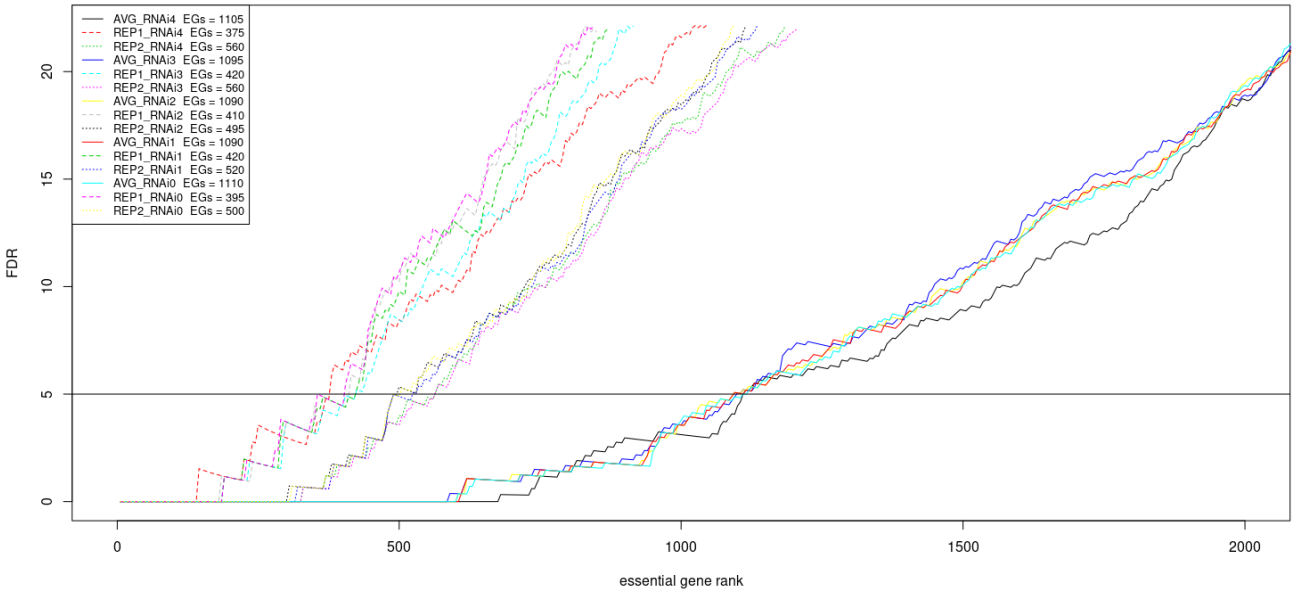


FIGURE A.1: False Discovery Rate of highly expressed genes according to gene essentiality in RNAi screen

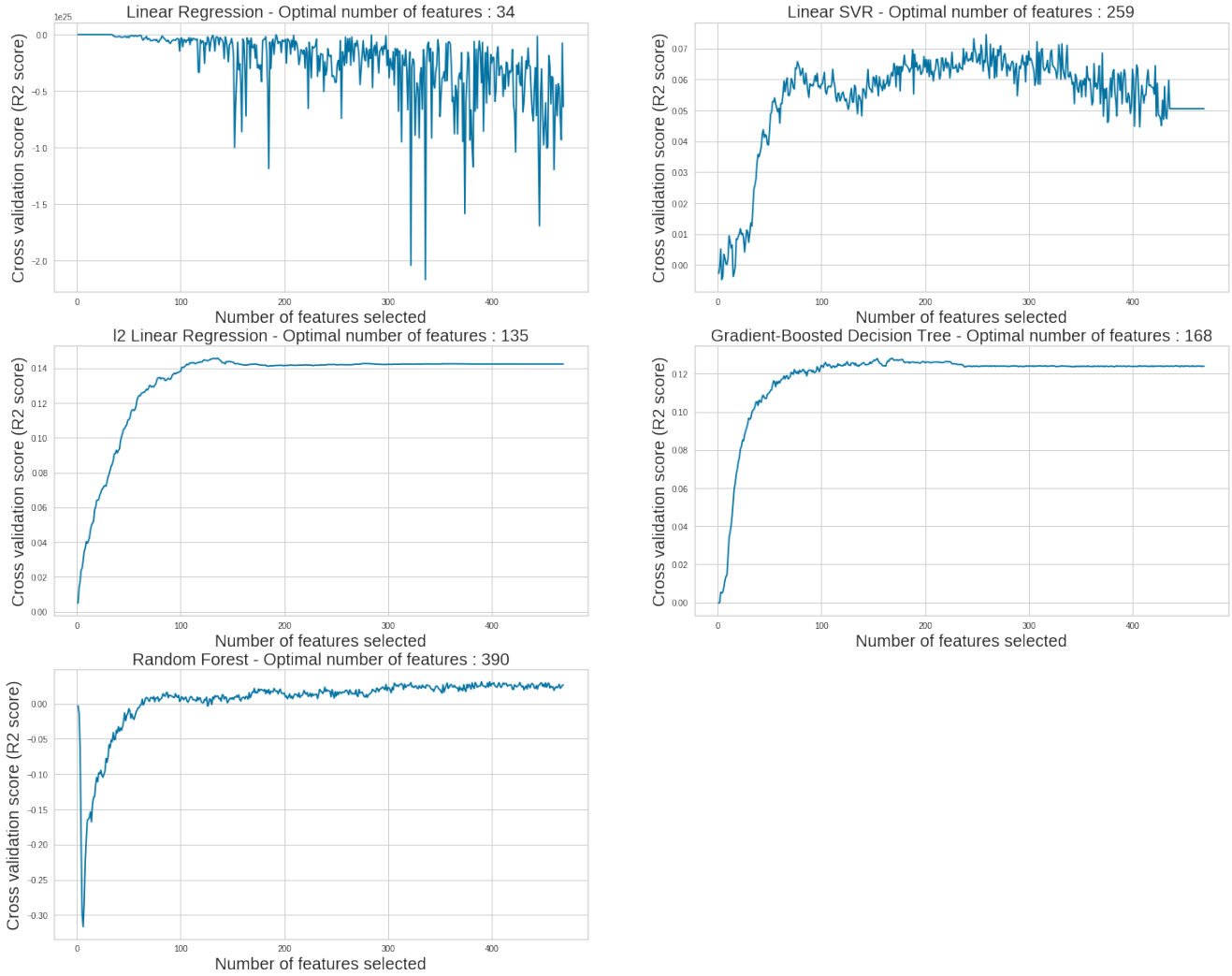


FIGURE A.2: RFECV to optimize the number of features for regression models of 23mer sgRNAs

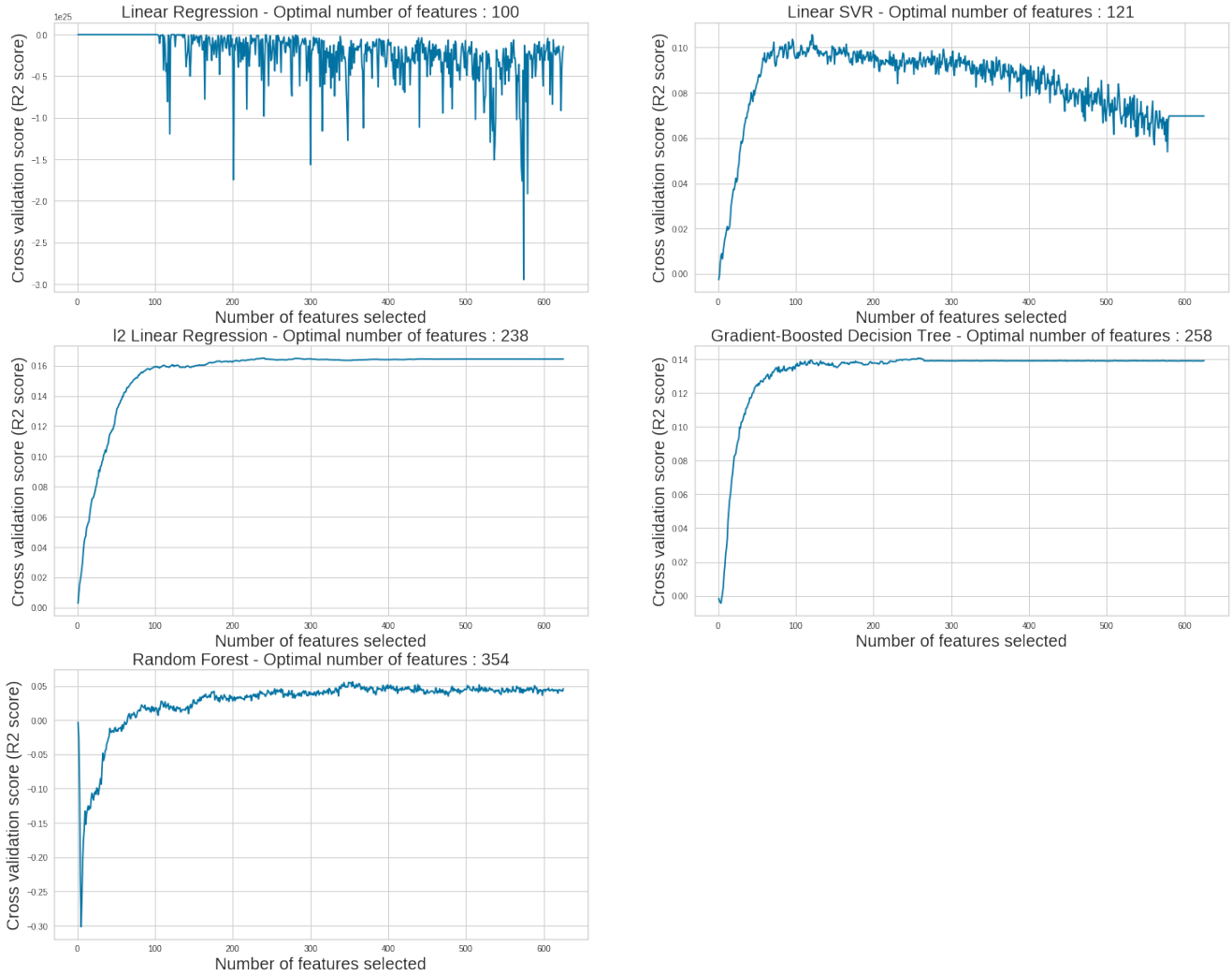


FIGURE A.3: RFECV to optimize the number of features for regression models of 30mer sgRNAs

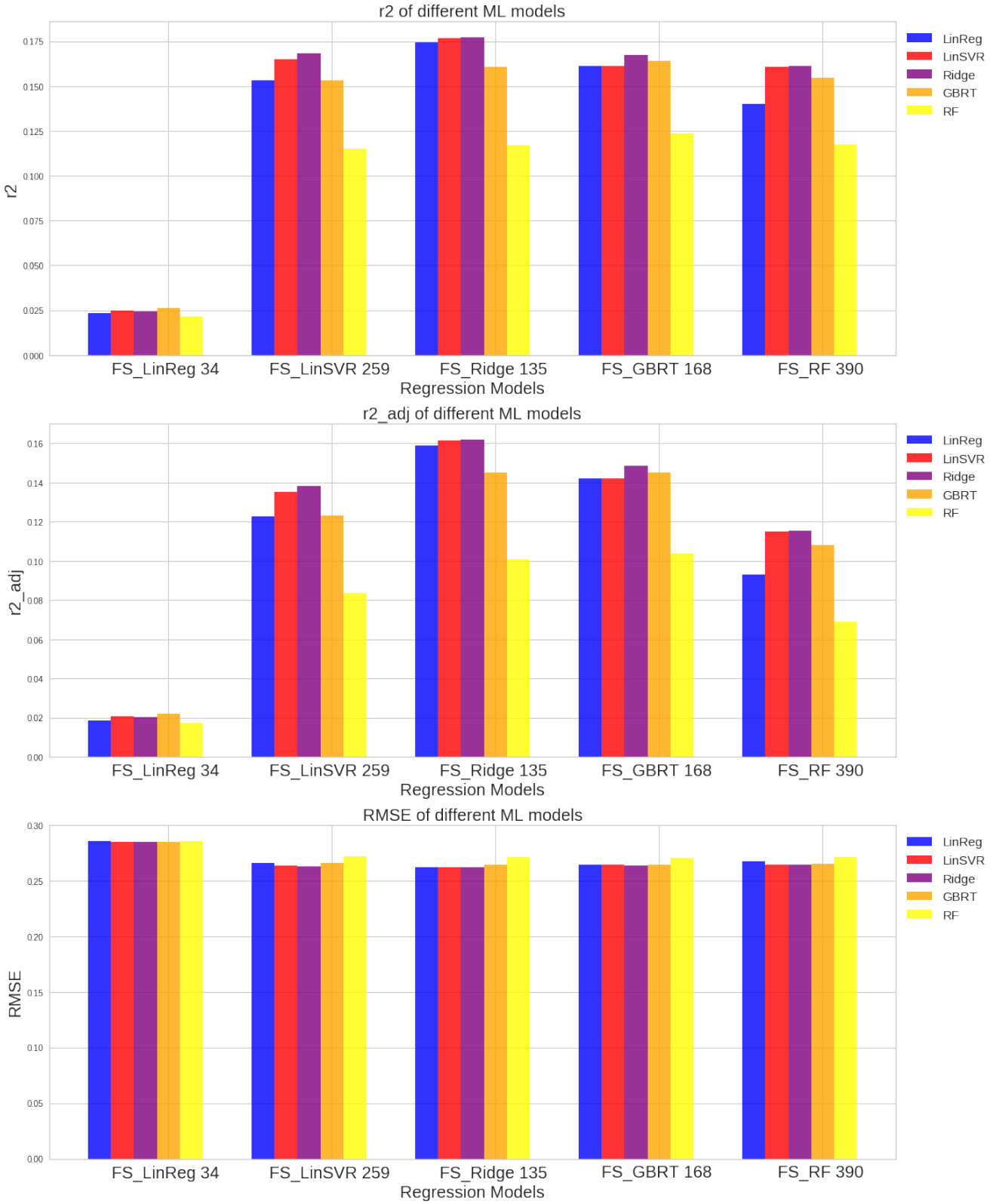


FIGURE A.4: Performance comparison of the regression models for 23mer sgRNA in each potentially optimal subsets of features

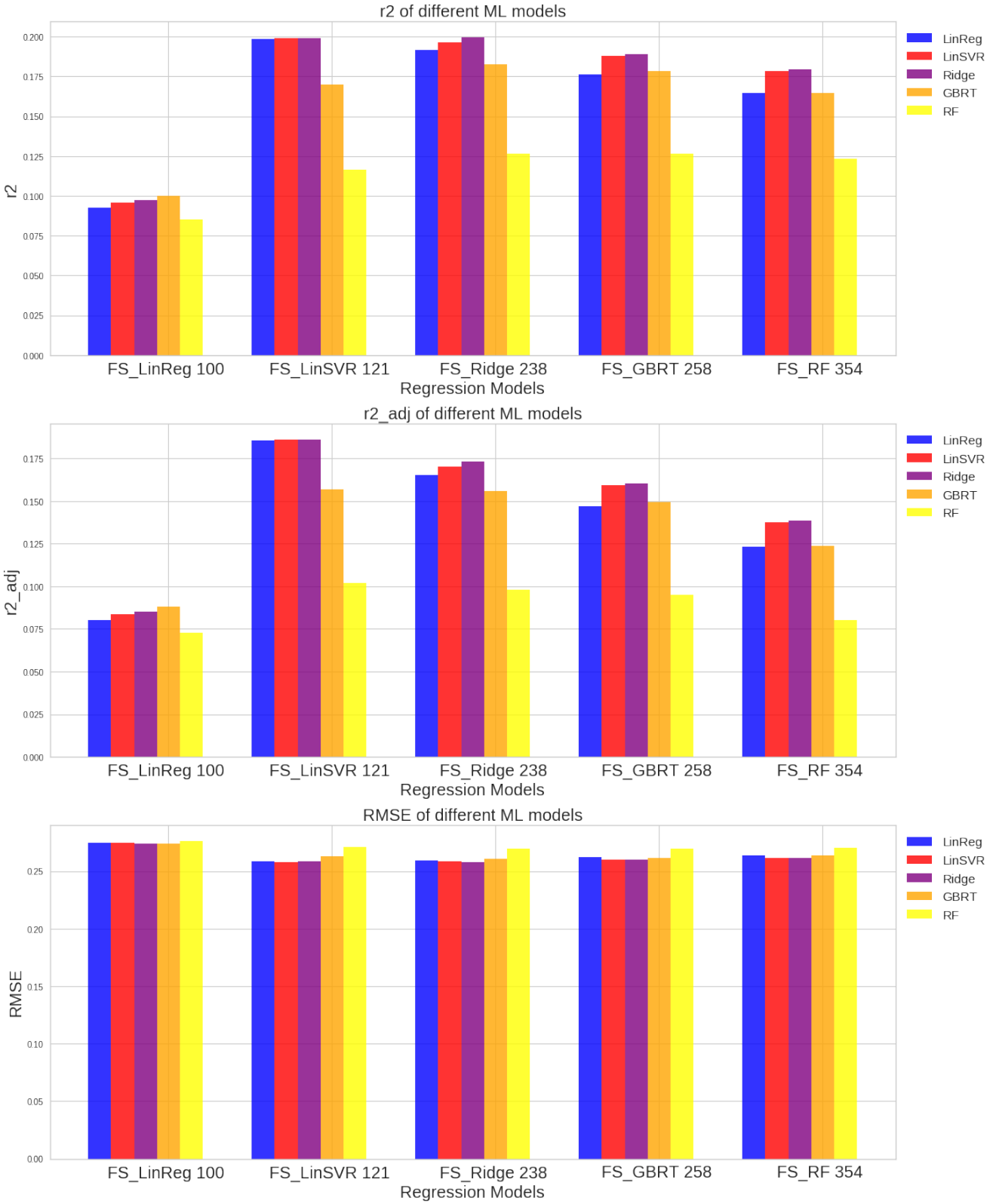


FIGURE A.5: Performance comparison of the regression models for 30mer sgRNA in each potentially optimal subsets of features

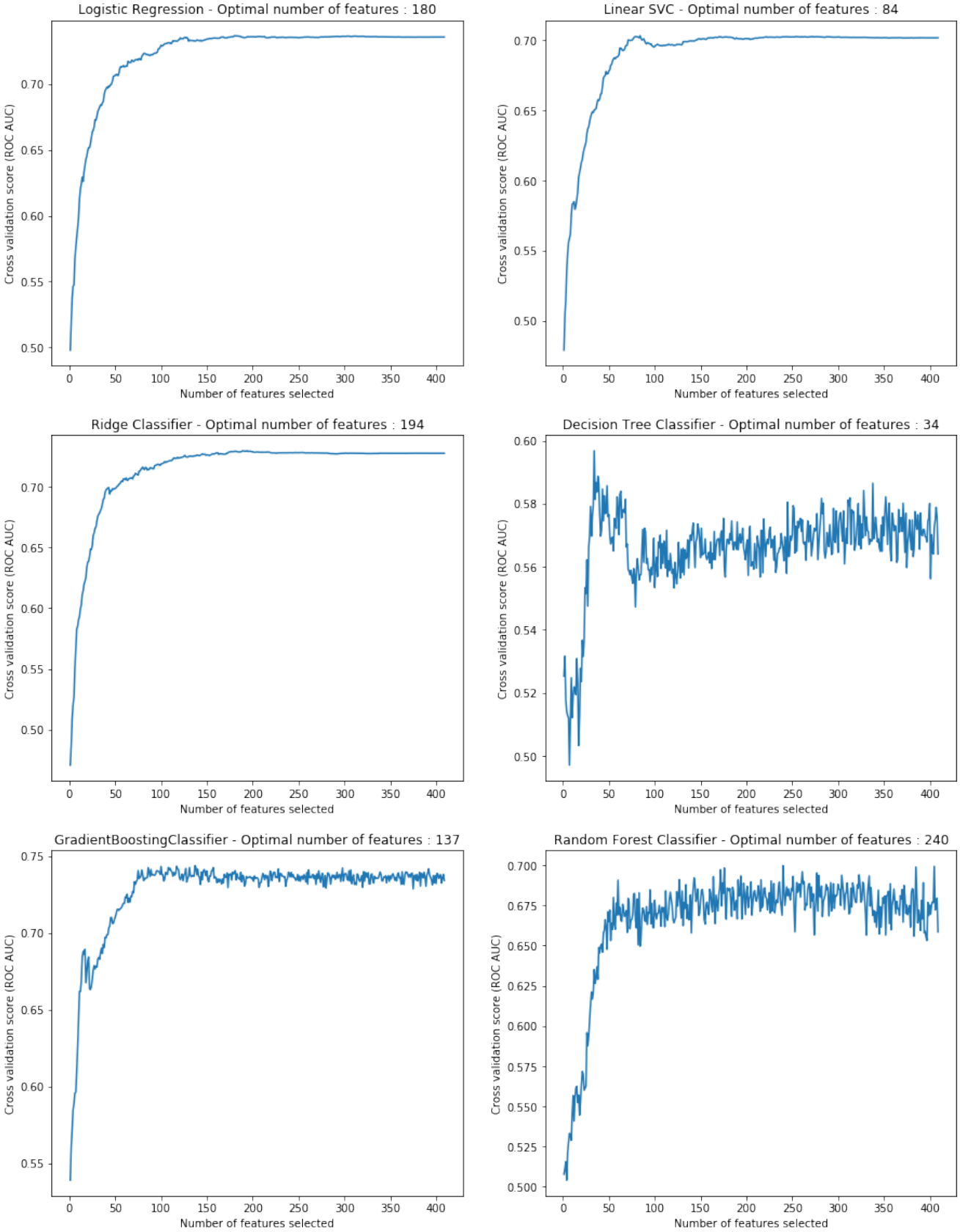


FIGURE A.6: RFECV to optimize the number of features for classification models of 20mer sgRNAs

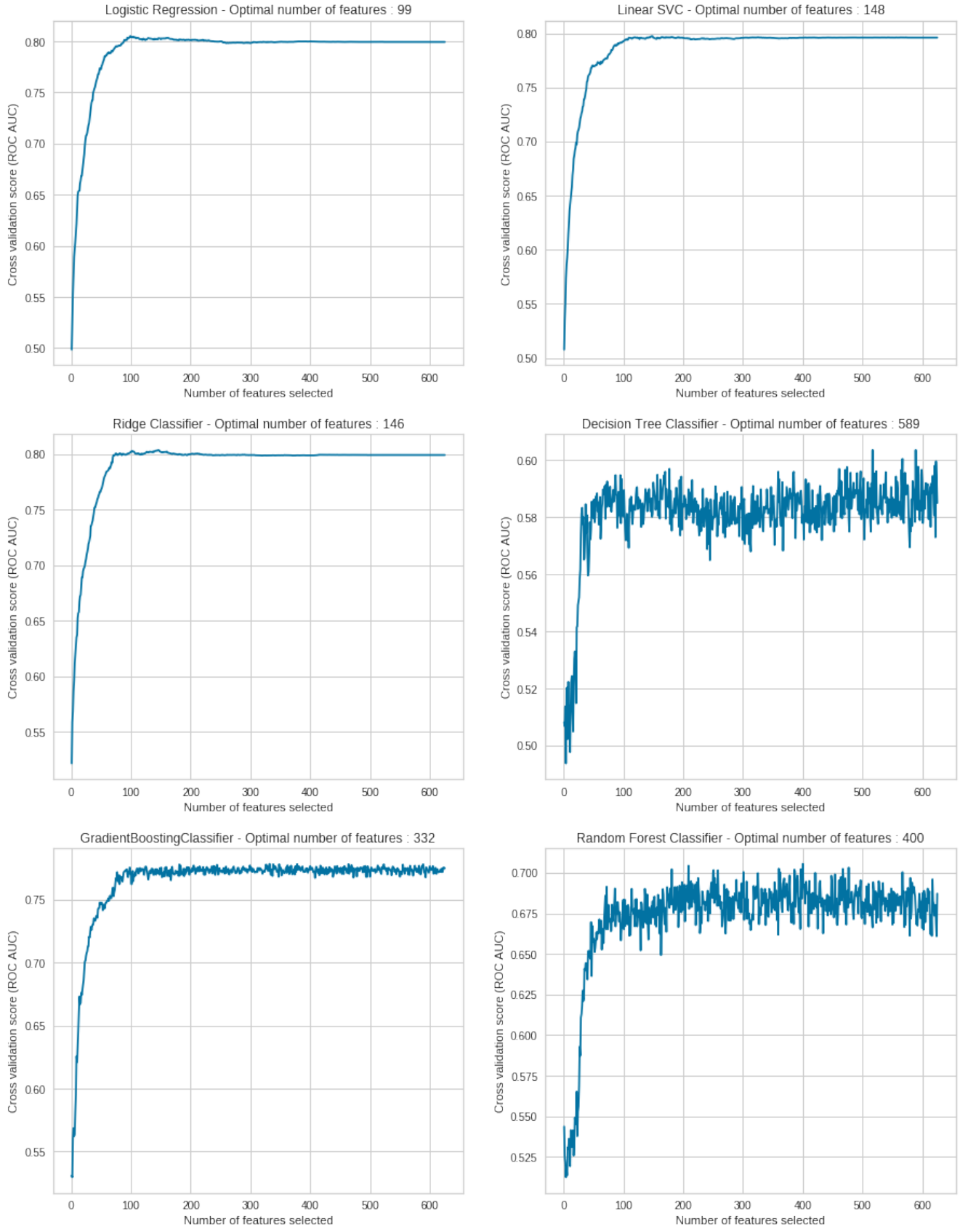


FIGURE A.7: RFECV to optimize the number of features for classification models of 30mer sgRNAs

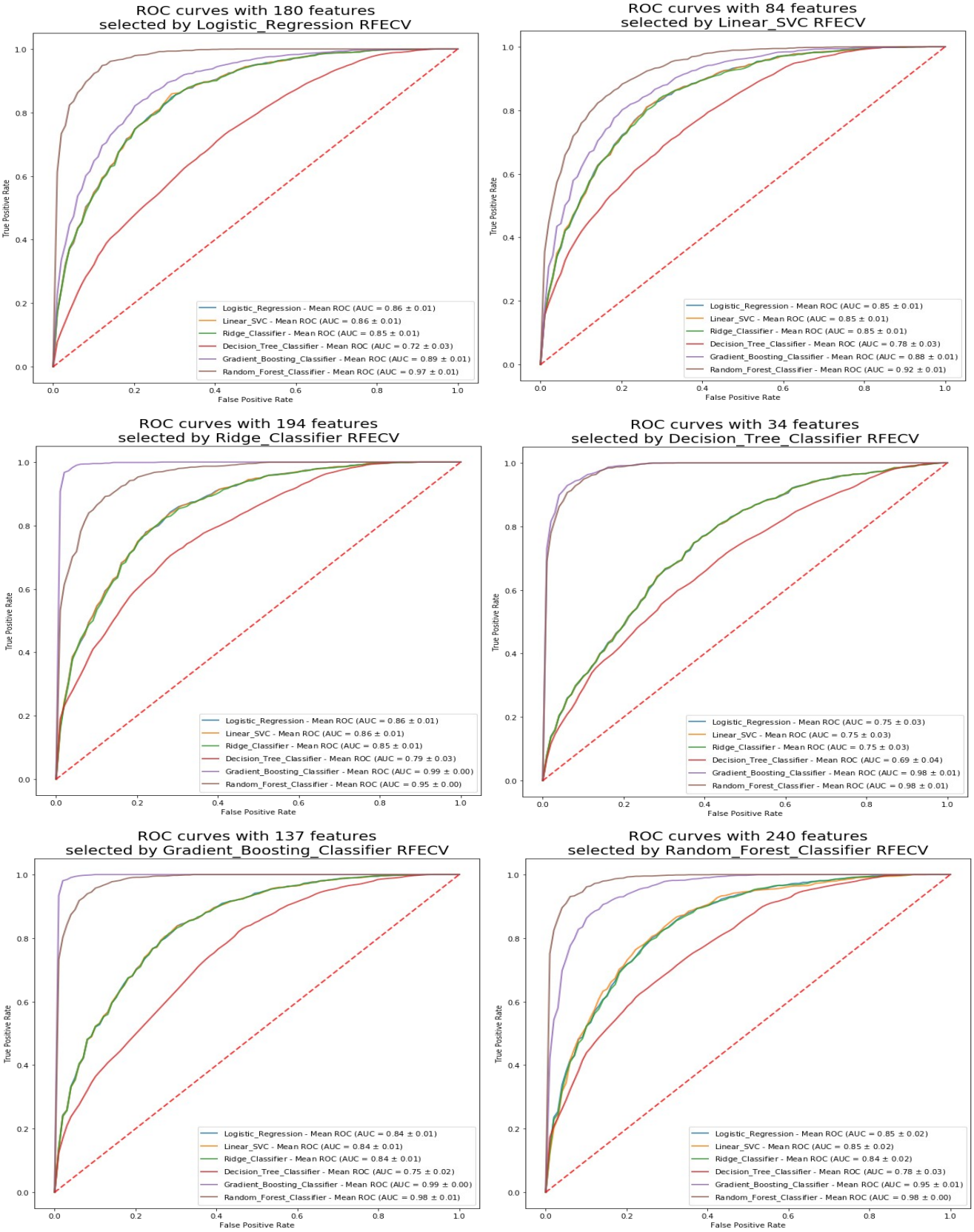


FIGURE A.8: Performance comparison of the classification models for 20mer sgRNA in each potentially optimal subsets of features

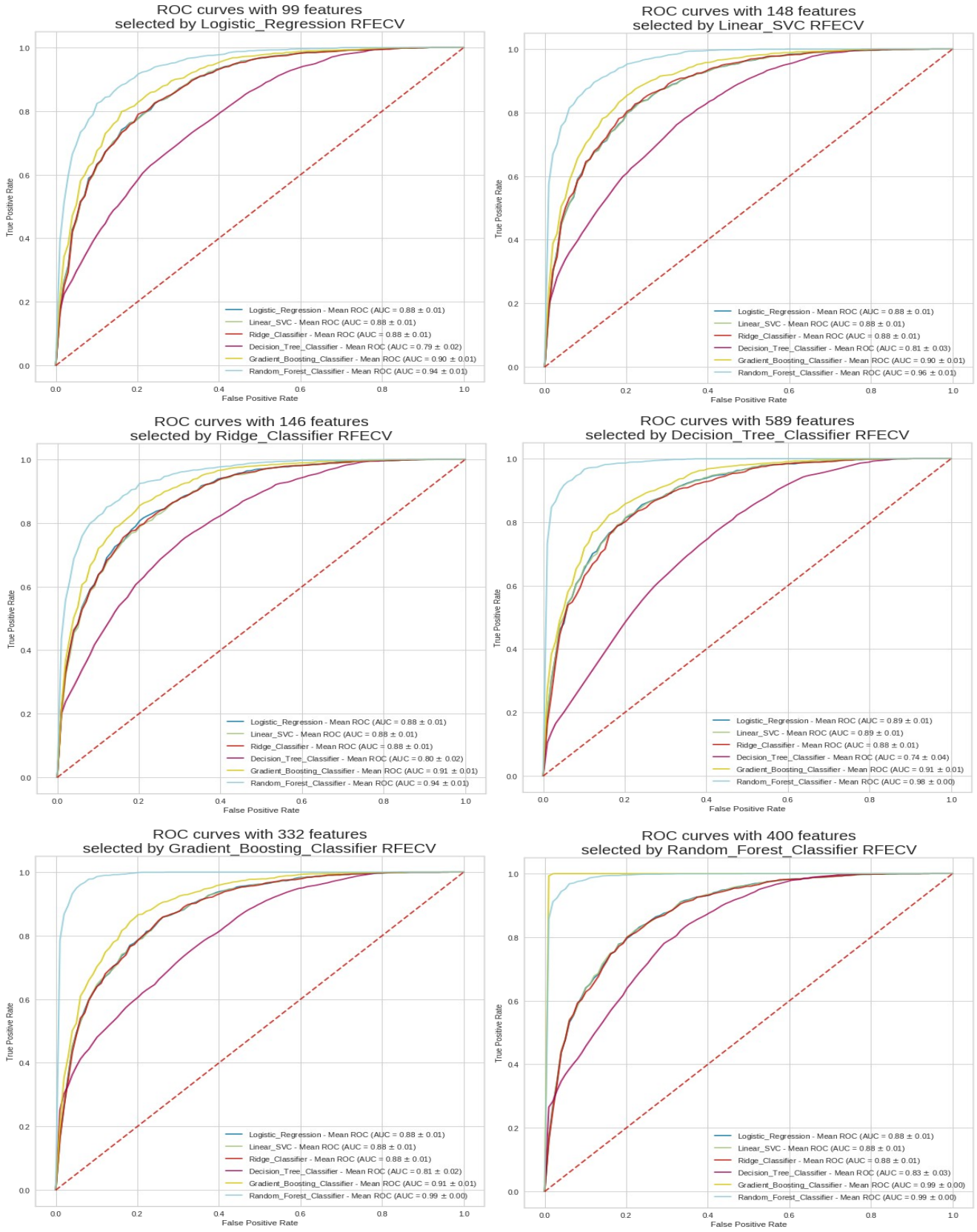


FIGURE A.9: Performance comparison of the classification models for 30mer sgRNA in each potentially optimal subsets of features

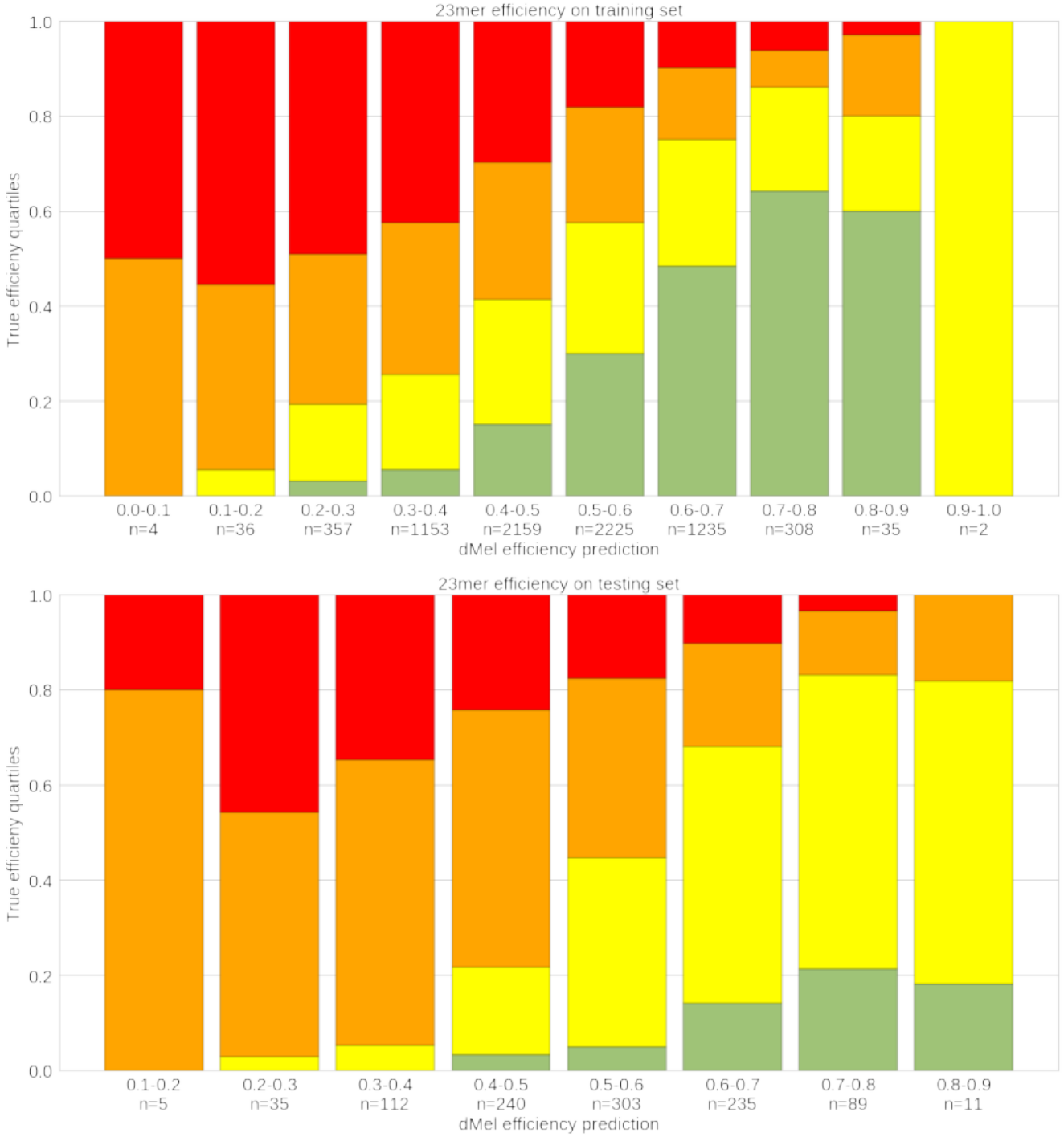


FIGURE A.10: Performance analysis of the regression model predictions for 23mer sgRNAs. red = 0.0 - 0.25 efficiency, orange = 0.25 - 0.50 efficiency, yellow = 0.50 - 0.75 efficiency, green = 0.75 - 1.0 efficiency.

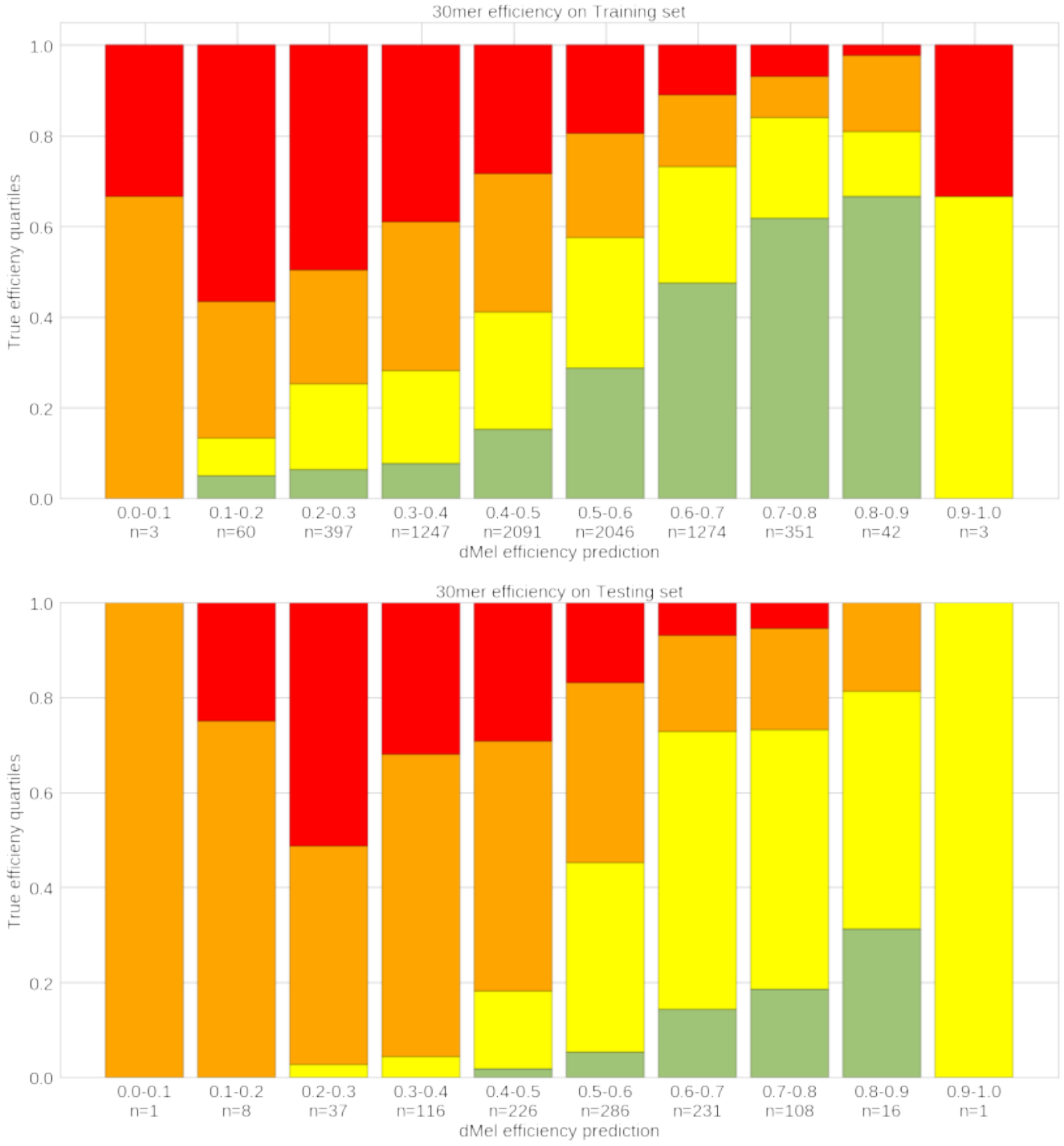


FIGURE A.11: Performance analysis of the regression model predictions for 30mer sgRNAs. red = 0.0 - 0.25 efficiency, orange = 0.25 - 0.50 efficiency, yellow = 0.50 - 0.75 efficiency, green = 0.75 - 1.0 efficiency.

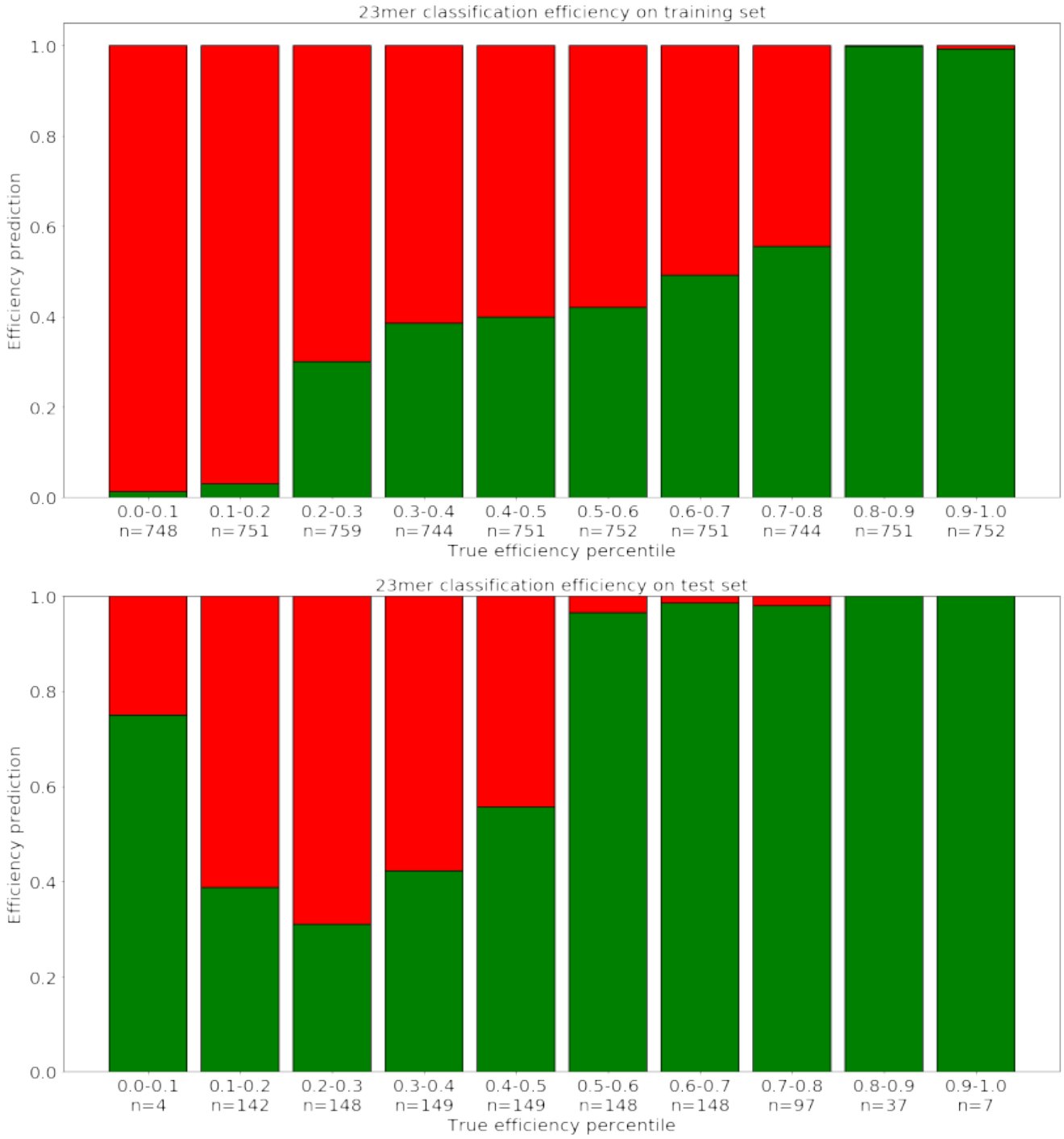


FIGURE A.12: Performance analysis of the classification model predictions for 23mer sgRNAs. red = 'low predicted efficiency. green = 'high predicted efficiency'.

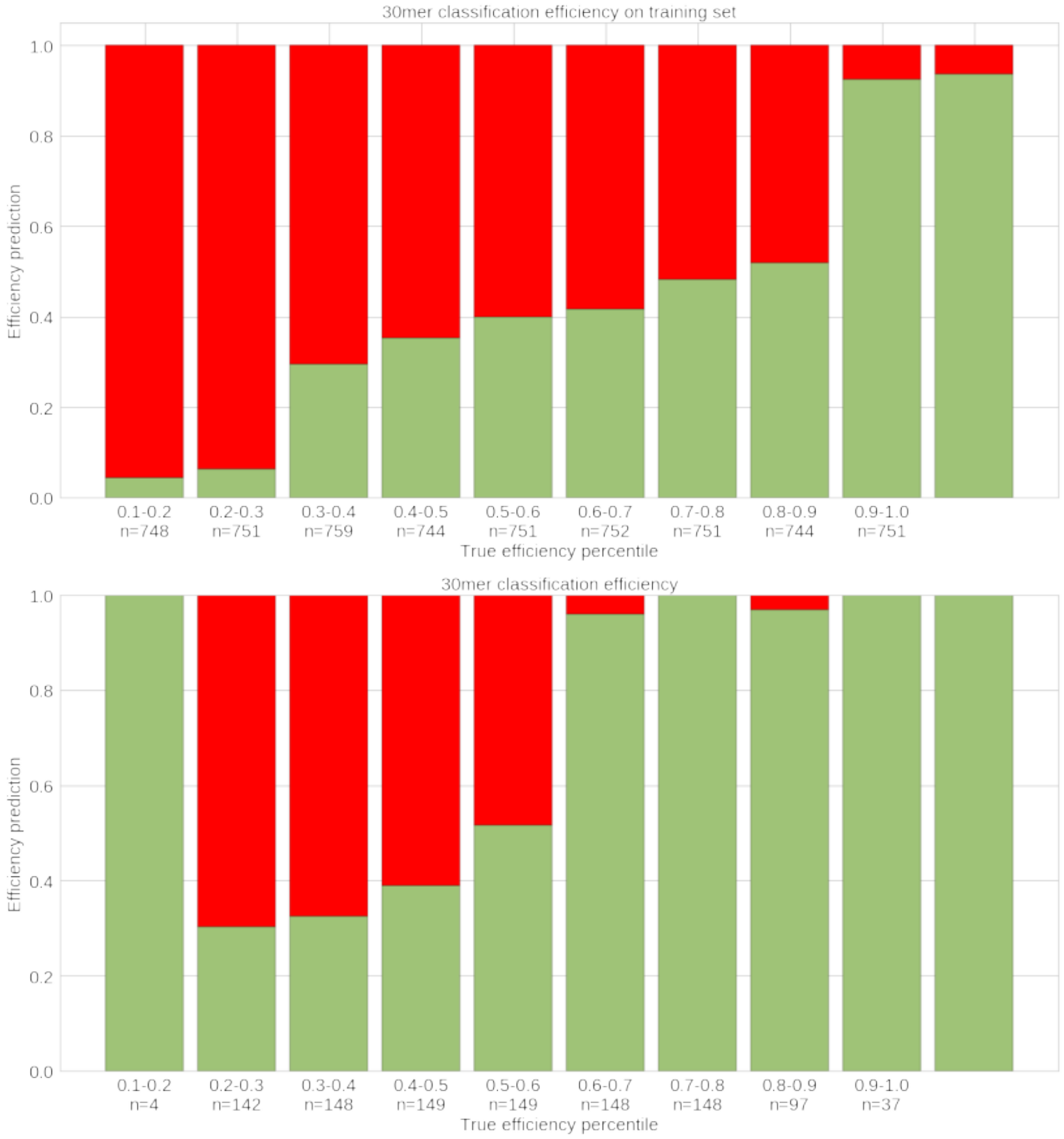


FIGURE A.13: Performance analysis of the classification model predictions for 30mer sgRNAs. red = 'low predicted efficiency. green = 'high predicted efficiency'.

Appendix B

Python Code for Dmel-sgRNA-Efficiency-Prediction tool

```
#!/usr/bin/env python
#-*- coding : utfs -*-

"""
Author: Pierre Merckaert
Contact: merckaert.pierre@gmail.com
Date: 06/2018

Description: Calculates the sgRNA efficiency score for a given 20, 23
    or 30-mer sgRNA or a csv file containing all sgRNA to predict.
Input: 20, 23 or 30mer sequence(s) : 20mer sgRNA + PAM (+ context
   sequence), NNNN[20mer sgRNA sequence]NGGNNN
Output: efficiency score(s)
"""

import os
import utils.util as util

if __name__ == '__main__':
        args = util.get_parser().parse_args()
        path = os.path.dirname(os.path.realpath(__file__)) #path of
           the directory of this python script

        #Launch the R preprocessing pipeline depending of the input
           arguments
        print_res = util.Launch_R_Preprocessing(args,path)

        #launch the Python pipeline to retrieve the sgRNAs sequences,
            their features and the Machine Learning model
        gRNAs_seq, gRNAs_param_REG, gRNAs_param_CLASS, ML_Model_REG,
           ML_Model_CLASS = util.PythonPreprocessing(path)

        #Predictions!
        scores_REG = ML_Model_REG.predict(gRNAs_param_REG)
        scores_CLASS = ML_Model_CLASS.predict(gRNAs_param_CLASS)
```

```
        #Output and save results depending on arguments input and
           processing outputs
        util.Output_Results(gRNAs_seq, gRNAs_param_REG, scores_REG,
           gRNAs_param_CLASS, scores_CLASS, args, path)

        #print prediction score in the terminal
        if print_res :
                if args.bin != "no":
                        print('Predicted efficiency score [0:1] for',
                            args.seq.upper(),'= ',scores_REG, '(the
                           higher the better)')
                        if scores_CLASS == 0:
                                print('The classification model
                                   predicts that ', args.seq.upper(),
                                   ' will NOT be an effective sgRNA.'
                                   )
                        else : print('The classification model
                           predicts that ', args.seq.upper(),' will
                           be an effective sgRNA.')
                else : print('Predicted efficiency score [0:1] for',
                   args.seq.upper(),'= ', scores_REG, '(the higher
                   the better)')
```

```
#!/usr/bin/env python
#-*- coding : utfs -*-

"""
Author: Pierre Merckaert
Contact: merckaert.pierre@gmail.com
Date: 06/2018
"""

import csv, argparse, os, sys, subprocess, pickle, sklearn, re
from sklearn.preprocessing import StandardScaler
import pandas as pd
import numpy as np
from collections import OrderedDict

# Description: get the input arguments and sets up the --help
# Output: parsed arguments
def get_parser():
```

```
        parser = argparse.ArgumentParser(description='''Calculates
           the sgRNA efficiency score for a given 20, 23 or 30-mer
           sgRNA OR a csv file containing all sgRNA to predict. The
           regression score ranges between 0 and 1 (The higher the
           better the sgRNA efficiency is predicted to be). the
           classification score is either 0 or 1 for poor and high
           efficiency respectively.''', epilog="")
        parser.add_argument('--seq',
                type=str,
                help='''20mer : [20mer sgRNA sequence]\n
                 23mer : [20mer sgRNA sequence]NGG\n
                 30mer : NNNN[20mer sgRNA sequence]NGGNNN''')
        parser.add_argument('--csv', type=argparse.FileType('r'),
                help='''csv file containing all sgRNA to predict
                   under a column header.\n
                Format : Comma-delimited csv file with the list of
                   20, 23 or 30mer sgRNAs in the first column. \n
                A header row is required.''')
        parser.add_argument('--out', type=str,
                help='''Path and/or Name of the predictions output
                   csv file. \n
                Default is PATH_TO_FOLDER/sgRNA_efficiency_prediciton
                   /sgRNA_predictions.csv''')
        parser.add_argument('--bin', type=str, default='no',
                help='''Choose with "yes" or "no" (default "no")
                   whether or not to include binary classification
                   predictions of the sgRNA efficiency in the output
                   csv file''')
        return(parser)

# Description: Run the R pipeline to extract the features from the
   input sequences
# Input: f_input is the csv file or the sgRNA sequence          path
   is the path of the current script
# Output: outputs the extracted features in /Rpreprocessing/
   R_Featurized_sgRNA.csv
def Rpreprocessing(f_input, path):
        command = 'Rscript'
        path2script = str(path)+'/Rpreprocessing/
           sgRNA_Input_Processing.R'
        cmd = [command, path2script, f_input]
        subprocess.run(cmd)

# Description: open a given csv file and appends its rows to an array
# Input: csv file of the extracted features from the R pipeline
# Output: array of the extracted features from the R pipeline
def csvParser(processed_sgRNAs):
```

```
        try:
                with open(processed_sgRNAs) as f:
                        file = csv.reader(f,delimiter=',')
                        gRNAs = []
                        for row in file:
                                gRNA = row
                                gRNAs.append(gRNA)
        except:
                raise Exception("could not find input stored to file
                   %s" % processed_sgRNAs)

        return(gRNAs)

# Description: Python pipeline to load the ML model and create the
   dataset for the efficiency prediction
# Input: path of the current file
# Output: array of the sgRNAs, features of these sgRNAs and the ML
   model to make the efficiency predictions
def PythonPreprocessing(path):
        # Opens the extracted features of the R Pipeline
        try :
                df_gRNAs = np.array(csvParser(str(path)+'/
                   Rpreprocessing/R_Featurized_sgRNA.csv'), dtype='<
                   U32')
        except :
                raise Exception("could not find Featurized sgRNAs
                   stored to file %s" % df_gRNAs)

        if df_gRNAs.shape[1] == 470 : #number of features extracted
           from 23mer sgRNAs
                #Features of the sequences
                gRNAs_param = np.array(df_gRNAs[1:,1:], dtype='
                   float64')

                model_REG_file = str(path)+'/utils/23
                   mer_135FS_Ridge_REGmodel.pickle'
                model_CLASS_file = str(path)+'/utils/23
                   mer_120FS_GBC_BINmodel.pickle'

                gRNAs_seq = df_gRNAs[1:,0] #23mer sequences

        elif df_gRNAs.shape[1] == 410 : #number of features extracted
            from 20mer sgRNAs
                #Features of the sequences
                gRNAs_param = np.array(df_gRNAs[1:,1:], dtype='
                   float64')
```

```
            model_REG_file = str(path)+'/utils/20
               mer_102FS_Ridge_REGModel.pickle'
            model_CLASS_file = str(path)+'/utils/20
               mer_137FS_GBC_BINmodel.pickle'

            gRNAs_seq = df_gRNAs[1:,0] #20mer sequences

    elif df_gRNAs.shape[1] == 627 : #number of features extracted
        from 30mer sgRNAs
            #Features of the sequences
            gRNAs_param = np.array(df_gRNAs[1:,2:], dtype='
               float64')

            model_REG_file = str(path)+'/utils/30
               mer_121FS_LinSVR_REGmodel.pickle'
            model_CLASS_file = str(path)+'/utils/30
               mer_400FS_GBC_BINmodel.pickle'

            gRNAs_seq = df_gRNAs[1:,0:2]     #23 and 30mer
               sequences
    else :
            raise Exception('Error : wrong R pipeline output
               dimension')

    #Standard Scaling of the non-binary features
    sc = StandardScaler()
    gRNAs_param[:,0:25] = sc.fit_transform(gRNAs_param[:,0:25])

    #Load regression model
    try:
            pickle_in_REG = open(model_REG_file,"rb")
            p_load_REG = pickle.load(pickle_in_REG)
            ML_Model_REG = p_load_REG['model']
            idx_REG = p_load_REG['df_indexes']
    except:
        raise Exception("could not find Regression model stored
           to file %s" % model_REG_file)

    #Load classification model
    try:
            pickle_in_CLASS = open(model_CLASS_file,"rb")
            p_load_CLASS = pickle.load(pickle_in_CLASS)
            ML_Model_CLASS = p_load_CLASS['model']
            idx_CLASS = p_load_CLASS['df_indexes']
    except:
        raise Exception("could not find classification model
           stored to file %s" % model_CLASS_file)
```

```
        gRNAs_param_REG = gRNAs_param[:,idx_REG] #Keep only the
           relevant features
        gRNAs_param_CLASS = gRNAs_param[:,idx_CLASS]

        return(gRNAs_seq, gRNAs_param_REG, gRNAs_param_CLASS,
           ML_Model_REG, ML_Model_CLASS)


# Description: run the R pipeline depending on the input given
# Input: arguments given to the script and path of the current file
# Output: csv of the featurized sgRNAs and if the prediction scores
   should be written in the terminal
def Launch_R_Preprocessing(args,path):
        assert ((args.csv is not None and args.seq is None) or (args.
           csv is None and args.seq is not None)), "you have to
           specify either 20, 23 or 30mer sgRNA sequence OR a csv
           file (see help)"
        if args.csv != None :
                # Extract features from input csv file
                Rpreprocessing(os.path.realpath(args.csv.name), path)
                print_res = False
        elif ((len(args.seq)==30) | (len(args.seq)==23) | (len(args.
           seq)==20)) & (re.match(r"^[ATGCatgc]*$", args.seq)!=None):
                # Extract features from input sequence
                Rpreprocessing(args.seq.upper(), path)
                print_res = True #print prediction score in the
                   terminal
        else :
                raise Exception("wrong input (see --help)")

        return(print_res)


# Description: Outputs the sgRNAs prediction scores in csv files
# Input:
        #gRNAs_param_REG : values of the relevant features for
           regression model
        #gRNAs_seq : list of the sgRNAs to predict
        #scores : prediction scores
# Output: save the prediction scores to csv file
def Output_Results(gRNAs_seq, gRNAs_param_REG, scores_REG,
   gRNAs_param_CLASS, scores_CLASS, args, path):
        if gRNAs_param_REG.shape[1] == 135 : #shape of the dataframe
           after feature selection for 23mer sgRNAs
                #Change the output if specified
                if args.out != None :
```

```
                output = args.out
        else :
                output = path+"/23mer_sgRNA_predictions.csv"
        #outputs the scores in a dataframe and save
        try :
                if args.bin == "no":
                        pd.DataFrame(OrderedDict({'gRNA_23mer
                           ' : gRNAs_seq, 'Efficiency
                           prediction' : scores_REG})).to_csv
                           (output, index=False)
                else:
                        pd.DataFrame(OrderedDict({'gRNA_23mer
                           ' : gRNAs_seq, 'Efficiency
                           prediction' : scores_REG, 'Binary
                           score' : scores_CLASS})).to_csv(
                           output, index=False)
        except:
                raise Exception("Error in saving prediction
                   results to %s. CSV file needed." % output)

elif gRNAs_param_REG.shape[1] == 102 : #shape of the
   dataframe after feature selection for 20mer sgRNAs
        #Change the output if specified
        if args.out != None :
                output = args.out
        else :
                output = path+"/20mer_sgRNA_predictions.csv"
        #outputs the scores in a dataframe and save
        try :
                if args.bin == "no":
                        pd.DataFrame(OrderedDict({'gRNA_20mer
                           ' : gRNAs_seq, 'Efficiency
                           prediction' : scores_REG})).to_csv
                           (output, index=False)
                else:
                        pd.DataFrame(OrderedDict({'gRNA_20mer
                           ' : gRNAs_seq, 'Efficiency
                           prediction' : scores_REG, 'Binary
                           score' : scores_CLASS})).to_csv(
                           output, index=False)
        except:
                raise Exception("Error in saving prediction
                   results to %s. CSV file needed." % output)

elif gRNAs_param_REG.shape[1] == 121 : #shape of the
   dataframe after feature selection for 30mer sgRNAs
        #Change the output if specified
```

```
            if args.out != None :
                    output = args.out
            else :
                    output = path+"/30mer_sgRNA_predictions.csv"
            #outputs the scores in a dataframe and save
            try :
                    if args.bin == "no":
                            pd.DataFrame(OrderedDict({'gRNA_23mer
                               ' : gRNAs_seq[:,1], 'gRNA_30mer' :
                                gRNAs_seq[:,0], 'Efficiency
                               prediction' : scores_REG})).to_csv
                               (output, index=False)
                    else:
                            pd.DataFrame(OrderedDict({'gRNA_23mer
                               ' : gRNAs_seq[:,1], 'gRNA_30mer' :
                                gRNAs_seq[:,0], 'Efficiency
                               prediction' : scores_REG, 'Binary
                               score' : scores_CLASS})).to_csv(
                               output, index=False)
            except:
                    raise Exception("Error in saving prediction
                       results to %s. CSV file needed." % output)
    else :
            raise Exception('Error : wrong R pipeline output
               dimension')
    print('DONE. Results exported to %s' % output)
```

Appendix C

Internship feedback

This internship in the Perrimon Lab at Harvard Medical School provided me with a tremendous amount of theoretical knowledge about the CRISPR/Cas9 system and Machine Learning. I learned more about the molecular mechanisms of CRISPR/Cas9, its applications in genetic screening, its current challenges and limitations in both the research and therapeutic contexts.
Regarding Machine Learning, I consolidated what I had been taught in class and now feel very confident in my abilities to conduct supervised machine learning projects.
For the technical aspect, my programming skills definitely improved in R and Python. In R I now manage and operate all sorts of dataframe operations with ease. In Python I now have a solid grasp of the numpy, pandas, scikit-learn and matplotlib libraries.
A remark that I have is that none of my mentors were familiar with Machine Learning, which is I think a good and a bad thing. In one hand, I had a strong sense of responsibility, autonomy and I learned a lot by conducting my own research. In another hand, I think that I would have been less stuck, have made less mistakes and would have probably learned more techniques and methodologies (e.g. Deep Learning) had I had a mentor versed in Machine Learning.

The global context of the internship surpassed by far all of my expectations. Being immersed in the prestigious Harvard University in the very dynamic city of Boston not only greatly improved my English skills but also allowed me to make lots of contacts and attend a lot of events relevant to the path I wish to pursue. I had the opportunity to meet with John Doench, one of the main developer of Azimuth, from the Broad Institute who gave me great feedback on how to conduct my Machine Learning analysis. Boston being one of the largest hub of biotech companies I was also able to attend lots of networking events and conferences such as CRISPRcon or the World Preclinical Congress where I met with Merck and Synthego (a late-stage startup whose CRISPR sgRNA design tools are very impressive) executives.

Machine Learning improvement of CRISPR/Cas9 sgRNA design is exactly the internship subject I was looking for, I am extremely grateful for this opportunity. It consolidates my thoughts that this field of research might be a career path suited for me.