Eidesstattliche Versicherung

(Affidavit)

Palguna, Supritha

Name, Vorname
(surname, first name)

229593

Matrikelnummer
(student ID number)

☐ Bachelorarbeit
(Bachelor's thesis)

■ Masterarbeit
(Master's thesis)

Titel
(Title)

Feature generation for classification of tumor development for mice fed with Western diet

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Dortmund, 01-12-2023

Ort, Datum
(place, date)

Unterschrift
(signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Dortmund, 01-12-2023

Ort, Datum
(place, date)

Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**

TU Dortmund

Master Thesis Report

Feature generation for Classification of tumour development for Mice fed with Western diet

First Supervisor: Prof. Dr. Jörg Rahnenführer
Second Supervisor: Dr. Franziska Kappenberg

Author: Supritha Palguna

December 1, 2023

Contents

1 Introduction

Cancer remains a significant challenge to human health, and the complex molecular mechanisms underlying its development are a subject of extensive research. This project focuses on studying mice that were fed two different types of diets: Standard diet (SD) and Western diet (WD) or High-fat diet (HFD). We explore gene expression analysis and tumor classification, aiming to shed light on the critical genetic factors contributing to the development of a tumor. The main objective of this research is to accurately classify tumor development in mice fed with a Western diet.

This project revolves around gene expression data, which provides information about how genes behave in response to different diets. The goal here is to uncover the genes which might be responsible for tumor development in mice when they consume a Western diet. To achieve this, we employ a comprehensive methodology that involves several stages, including data preprocessing, feature selection, machine learning based classification, and filtering methods.

At first, the gene expression data is cleaned and prepared to ensure that it is suitable for analysis. The DESeq2 package in R is utilized to perform the initial data preprocessing steps, and Principal Component Analysis (PCA) is employed to gain insights into the data structure. Later, univariate analysis is performed to assess predictive capability of genes.

One of the key aspects of our analysis is the selection of relevant genes. We apply five distinct feature selection methods. The first approach involves identifying genes that exhibit significant upregulation after specific time points. Similarly, we also identify genes that display significant downregulation. Furthermore, we utilize dose-response modeling technique to uncover genes with distinctive patterns over time. Lastly, we select the top 500 genes with the greatest variability specifically. It is important to note that all of these gene selection methods are based on the standard diet data. The selected gene sets are then merged with information about tumor development, creating datasets that include a binary classification label (1 for tumor, 0 for no tumor).

Subsequently, four different classification methods are employed, namely Random Forest, Support Vector Machine (SVM) with linear kernel, SVM with radial kernel, and Penalized Logistic Regression, to distinguish genes associated with tumor development.

Lastly, four filter methods, including variance, permutation, impurity, and JMIM filters, are applied to all genes without prior feature selection, as well as to the genes selected from feature selection.

The key findings of this research will help us understand which genes are linked to tumor development influenced by diet. By effectively categorizing these genes, we aim to enhance our knowledge about tumour development. This research could offer essential knowledge for future research in cancer studies.

In Section 2, a comprehensive overview of the dataset is provided, offering insights into how the data was gathered, its quality, and expressing the primary objective of the project. This section lays the foundation by setting the context for the subsequent analysis. Section 3 is dedicated to the demonstration of various statistical methods employed. These methods include Principle components analysis, DESeq2, the Four-parameter log-logistic model (4pLL), Random Forest, Support Vector Machines, Penalized Logistic Regression, filter methods, hyperparameter optimization and performance measures. Section 4 presents the application of these statistical methods for a detailed analysis of the dataset. Finally, the synthesis of our discoveries, significant findings, and key takeaways is encapsulated in the concluding section.

2 Problem statement

This section thoroughly presents the dataset and the objective of the project.

2.1 Description of Dataset

The dataset utilized in this project is obtained from a study 'Spatio-Temporal Multi-scale Analysis of Western Diet-Fed Mice' (Ghallab et al., 2021). It consists of two primary files - ColumnData summarizing key variables, and CountData containing gene expression read counts.

The ColumnData file includes the variables *Weeks*, *Diet* and *Mouse*. *Week* is an ordinal variable and represents the time at which the samples are collected and is crucial for understanding how gene expression changes over time. Samples are collected at nine different time points, which include weeks 3, 6, 12, 18, 24, 30, 36, 42, and 48. *Diet* variable is nominal, categorizing mice into two diet groups, a standard diet group and a Western diet group. *Mouse* represents mouse identifiers (M1, M2, etc).

The CountData file comprises read counts of 35,727 total genes for each mouse sample. Overall, the data comprises a total of 79 samples, with 32 of them corresponding to mice fed with a standard diet and 47 to mice fed with a Western diet.

An additional tumor file contains information about the tumour development in mice. *Tumor* variable is binary (0 or 1). After the 30-week mark, a subset of 10 out of 22 mice developed tumors. This variable indicates whether a mouse developed a tumor (coded as 1) or not (coded as 0).

2.2 Project Objective

The primary objective in this project is the selection of genes exhibiting distinct behaviors over time. This is achieved by the selecting genes exhibiting significant upregulation or downregulation patterns over time, identifying genes with different expression patterns through dose-response modeling and selecting the top 500 genes with the highest variability. Notably, these methods are exclusively applied to mice that were fed a standard diet.

Furthermore, the central aim involves the classification of mice that developed tumors after week 30 and were subjected to a Western diet. To achieve this, several machine learning methods are employed, and their performances are systematically compared and evaluated. To ensure the robustness of our results, the machine learning models are validated by assessing their predictive accuracy and confusion matrix. This is crucial in determining the reliability of the classification results and in identifying the best method for this specific task.

3 Statistical methods

In this section, various statistical methods employed for analyzing the dataset are explained. All analyses and visualizations are conducted using the R software, version 4.2.2 (R Development Core Team, 2013).

DESeq2 (Love et al., 2014), *ranger* (Wright et al., 2023), *e1071* (Meyer et al., 2023), *mlr3* (Bischl et al., 2023), *mlr3tuning* (Becker et al., 2023), *mlr3learners* (Lang et al., 2023), *mlr3pipelines* (Binder et al., 2023), *mlr3filters* (Schratz et al., 2023), *praznik* (Kursa, 2022), *drc* (Ritz and Strebig, 2016), *dplyr* (Henry et al., 2023), *ggplot2* (Wickham et al., 2023) are some of the packages used in this project.

3.1 Principal Components Analysis

Principal Component Analysis (PCA) offers a strategy to diminish the dataset's dimensionality, containing numerous correlated variables, while retaining the majority of the variation.

Consider a dataset consisting of n observations and p interrelated variables, represented as $X = X_1, X_2, \ldots, X_p$. Assume the task is to visualize these n observations. Creating scatterplots for each pair of features ($\frac{p(p-1)}{2}$ plots) becomes impractical for large p. The goal is to find a low-dimensional representation that retains crucial information. PCA achieves this by identifying a subset of dimensions, the principal components, which maximize the variance in the data.

Each observation exists in a p-dimensional space, although not all dimensions contribute equally. PCA finds a limited number of dimensions that are maximally informative, measured by the variance along each dimension. Each dimension, or principal component, is a linear combination of the original features.

The first principal component Z_1 is a normalized linear combination of features,

$$Z_1 = \alpha_{11}X_1 + \alpha_{21}X_2 + \ldots + \alpha_{p1}X_p$$

Here, $\alpha_{11}, \alpha_{21}, \ldots, \alpha_{p1}$ are the principal axes forming an eigen vector $\alpha_1 = (\alpha_{11}, \alpha_{21}, \ldots, \alpha_{p1})^T$. The constraint $\sum_{j=1}^{p} \alpha_{j1}^2 = 1$ ensures normalization.

Computing the first principal component (PC) involves in finding a linear combination, $z_{i1} = \alpha_{11}x_{i1} + \alpha_{21}x_{i2} + \ldots + \alpha_{p1}x_{ip}$, that maximizes sample variance, subject to normal-

ization constraints. This is formulated as an optimization problem, (James et al., 2009, p. 505-507)

$$\text{maximize} \quad \frac{1}{n}\sum_{i=1}^{n} z_{i1}^2$$

$$\text{subject to the constraint,} \quad \sum_{j=1}^{p} \alpha_{j1}^2 = 1$$

Understanding how to compute principal components is crucial. Let us consider $\mathbf{\Sigma}$ as the covariance matrix of the vector X. This matrix has elements representing the covariance between the ith and jth variables when $i \neq j$, and the variance of the jth variable when $i = j$.

The principal components derived via PCA are constructed as linear combinations of the original input features. Specifically, the k-th principal component Z_k is defined by $Z_k = \alpha_k X$, where α_k is a vector of weights for each input feature X.

These weight vectors α_k are chosen as the eigenvectors of the data covariance matrix $\mathbf{\Sigma}$. The corresponding eigenvalues λ_k quantify the variance along the new component directions. By selecting α_k vectors with unit length ($\|\alpha_k\| = 1$), this directly makes the variance of Z_k equal to the eigenvalue λ_k. (Jolliffe, 1986, p. 1-3)

This formulation emphasizes that PCs are constructed based on the eigenvectors of the covariance matrix $\mathbf{\Sigma}$. The key insight is that each PC represents a linear combination of the original variables, and these combinations are ordered by the amount of variance they capture. The projection of the data onto these PCs results in a reduced-dimensional representation that retains the most significant information present in the original dataset.

3.2 DESeq2

DESeq2, developed by Love et al. (2014) and available as an R package, primarily serves the purpose of analyzing gene expression differences between various phenotypes or conditions. This tool is mainly utilized for RNA-Seq data analysis, specifically designed to handle the unique challenges that come with count data. It effectively addresses issues like the high variability seen in low counts, which often reflect genes with low expression levels.

DESeq2 assumes that the count data generated by RNA-seq experiments follows a negative binomial distribution. This distribution is well-suited for modeling count data, which is common in gene expression studies, as it allows for the representation of data with greater variance than the mean. The analysis begins with a matrix K, where K_{ij} represents the count of reads from a sequencing experiment mapped to gene i from sample j. These counts, K_{ij}, are modeled using the Negative Binomial distribution (Love et al., 2023),

$$K_{ij} \sim NB(mean = \mu_{ij}, dispersion = \phi_i)$$

Here, μ_{ij} is the mean which is split into two components: s_j (a normalization factor) and Q_{ij} (a quantity proportional to the proportion of cDNA fragments of gene i in sample j).

The normalization factor, s_j, is estimated to adjust for differences in sequencing depth between samples, ensuring fair sample-wise comparisons. DESeq2 calculates s_j using the Median-of-Ratios method to mitigate the impact of variations in sequencing depth. Normalization is crucial for removing biases and ensuring that data from different samples are comparable.

The estimation of the expression strength of gene i is conducted using a generalized linear model,

$$log_2(Q_{ij} = \sum_r x_{jr}\beta_{ir}),$$

where $log_2(Q_{ij})$ is related to the design matrix X, and it estimates $log2$ fold changes (β_{ir}) between conditions, reflecting the gene's overall expression strength. For differential expression analysis, typically, $r = 1$, comparing two conditions.

Proper analysis using DESeq2 relies on accurate gene-wise dispersion parameter estimates, ϕ_i, particularly for small sample sizes. This ϕ_i models the within-group variability. The method assumes genes with similar average expression will show comparable dispersion. After initial gene-specific dispersion estimation via maximum likelihood, a curve is fit to capture the relationship between dispersion and mean expression over genes. Then final dispersion values are obtained per gene using an empirical Bayes approach, shrinking the maximum likelihood estimates towards this curve. The degree of shrinkage thereby depends on sample size and distance to the global dispersion trend. With the Bayes strategy, estimates borrow information across genes to improve stabilization, especially benefiting genes with high uncertainty.

DESeq2 also handles the issue of highly variable expression for genes exhibiting low read counts. This high variance can lead to exaggerated log fold change estimates. DESeq2 provides three shrinkage estimators, with the default based on a Cauchy prior distribution for coefficient β_i (Love et al., 2023),

$$\beta_i \sim Cauchy(0, scale_i)$$

The prior distribution's scale parameter, denoted as $scale_i$, is determined through Maximum Likelihood Estimates ($\hat{\beta}_i$) and their corresponding standard errors. It is assumed that the estimates ($\hat{\beta}_i$) follow a normal distribution centered around their true values (β_i).

Following the model fitting and the application of shrinkage to obtain adjusted log fold change estimates, a hypothesis test is executed to evaluate the statistical significance of the estimate deviation from zero. DESeq2 employs a Wald test as the default hypothesis test, where a z-score (adjusted log fold change divided by the estimated standard error) is compared to a standard normal distribution. To account for multiple testing, a correction is applied, and the adjusted p-value is employed to ascertain whether a gene demonstrates differential expression.

3.3 Four-parameter log-logistic model (4pLL)

Dose-response analysis is often implemented using four-parameter log-logistic model. For a dose x and four parameters b, c, d, and e, 4pLL model is represented as,

$$f(x, (b, c, d, e)) = c + \frac{d - c}{1 + \exp(b(\log(x) - \log(e)))},$$

where d is the upper asymptote and c lower asmptote of the function. Parameter e, which must be positive, signifies the dose that generates a response midway between the upper d and c, denoted as the ED50. The parameter b reflects the slope of the dose-response curve at the ED50. The actual slope of the tangent of the log-logistic model function at e is given by (Ritz et al., 2015, p. 178),

$$\frac{-b}{(d - c)/(4e)}$$

This scaling factor, dependent on c, d, and e, converts parameter b into the slope, making it more interpretable.

The four-parameter log-logistic model can be parameterized differently as well. An alternative parameterization replaces e with the logarithm of ED50 (Ritz et al., 2015, p. 179), denoted as $\tilde{e}$, as a model parameter,

$$f(x, (b, c, d, \tilde{e})) = c + \frac{d - c}{(1 + \exp(b(\log(x) - \tilde{e})}$$

This version of the 4pLL prove useful for modeling dose-response relationships when dataset sizes are limited. With fewer than 15-20 samples, assuming normality on the log-transformed dose scale tends to be more reasonable than the original scale.

The fitting of the model is performed using drm() function from R-package drc (Ritz and Strebig, 2016). The curves are fitted by minimizing the sum of squared errors between the data points and the fitted function, assuming normal distribution of response values.

3.4 Cross-validation

Cross-validation is a data resampling technique essential which is used to evaluate the generalization capability of predictive models and also control the risk of overfitting. It shares similarities with the repeated random subsampling method but ensures that no two test sets overlap.

The k-fold cross-validation strategy partitions the initial dataset, $\mathbf{X}$ into k exclusive subsets, or folds (denoted as $X_1, X_2 \ldots X_k$), which are approximately equivalent in size. This method performs iterative training and testing of the model k times, with one fold reserved for testing in each iteration. During this process, the remaining folds collectively constitute the training set, ensuring an equal utilization of each sample for both training and testing. This approach provides a comprehensive and robust evaluation of the model's performance.

In the context of classification, the accuracy estimate is calculated by taking the total number of correct classifications across all iterations and subsequently dividing it by the overall number of tuples present in the initial dataset.

In stratified cross-validation it ensures that the class distribution within each fold closely mirrors that of the original dataset. This is specifically helpful when there is a situation of class imbalances. (Han et al., 2012, p. 371-372)

3.4.1 Nested cross-validation

Nested cross-validation is a methodology designed to assess model performance and fine-tune hyperparameters. This technique extends the conventional cross-validation process by incorporating an additional layer of validation. The outer loop involves dividing the dataset into multiple folds, with each fold serving as a test set during a specific iteration. Simultaneously, the inner loop, nested within the outer loop, fine-tunes the model's hyperparameters. While nested CV offers a robust evaluation of model performance, it comes at the expense of increased computational demands. (Wainer and Cawley, 2021)

3.5 Random Forest

Random Forest is a versatile and powerful machine learning method that combines decision trees to make accurate predictions. It is based on the idea of ensemble learning, which combines the results of multiple models to improve predictive accuracy and reduce overfitting.

Consider a random forest ensemble comprising of trees $T_1, T_2, \ldots, T_B$. Let Z be of bootstrap sample of size N, where each bootstrap sample represents a random subset of the original dataset with p features.

The Random Forest algorithm (Hastie et al., 2008, p. 588) can be outlined as follows:

1. Draw a bootstrap sample Z of size N from the dataset.
2. For each bootstrapped sample, randomly select m features from p, where $m \approx \sqrt{p}$.
3. For each bootstrapped sample and its corresponding random feature subset, grow a decision tree to its maximum depth without pruning.
4. At each node, identify the best split among the randomly selected features.
5. Repeat steps 1-4 for all B trees.

For a new observation x, each tree T_b predicts a class denoted as $C_b(x)$. The final prediction $C_{rf}^B(x)$ is determined through majority voting across all trees, expressed as:

$$C_{rf}^B(x) = \text{Majority Vote}\{C_b(x)\}_1^B$$

Random Forest can estimate its own generalization error through out-of-bag (OOB) samples. Each tree is constructed using a bootstrap sample, and the OOB samples

are the data points not included in that tree's training sample. The OOB error is calculated as the proportion of misclassified OOB samples in classification tasks or the mean squared error in regression tasks (Hastie et al., 2008, p. 592).

Random Forest is a robust and versatile ensemble learning method that is widely used for various machine learning tasks. It provides built-in mechanisms for controlling overfitting, handles noisy data well, and offers a natural way to estimate model performance.

3.6 Support Vector Machines

Support Vector Machines (SVM) is a powerful machine learning algorithm that plays a prominent role in classification and regression tasks. It works by finding the best line or boundary (hyperplane) that separates different classes of data points in space. The positioning of the hyperplane in such away that it maximizes the margin, which is defined as the distance between the hyperplane and the nearest data points from each class.

Let the training data be represented by X, which comprises of N points, $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$. Here, x_i is data point in a p-dimensional real space ($\mathbb{R}^p$) and y_i is the class variable which takes values in the set {-1, 1}. The hyperplane $f(x)$ is defined by the equation,

$$\{x : f(x) = x^T\beta + \beta_0 = 0\},$$

where x is an input vector, β is a weight vector and has a unit length and β_0 is a bias term.

The associated classification rule $G(x)$ signifies the signed distance from a point x to the hyperplane and is given by the equation,

$$G(x) = \text{sign}[x^T\beta + \beta_0].$$

If $G(x)$ is positive, the point is classified as one class; if negative, it belongs to the other class. Hence, the hyperplanes can be written as,

$$f(x) \geq +1 \quad \text{or} \quad f(x) \leq +1 \quad \text{with} \quad y_i = \pm 1, \quad \text{respectively}$$

The above inequalities can be combined into one equation as, $y_i f(x) \geq 1$ (James et al., 2009, p. 418).

The margin m, defined as $m = \frac{1}{||\beta||}$, can be maximized by minimizing $||\beta||$,

$$min_{\beta,\beta_0}||\beta||$$

$$\text{subject to }, \quad y_i(x^T\beta + \beta_0) > 0, i = 1, \ldots, N$$

Let $\xi = \xi_1, \xi_2, \ldots, \xi_N$ be slack variables. When the data is not linearly seperable, SVM uses these slack variables. It attempts to minimize misclassified data with cost variable C and using slack varaiables as (James et al., 2009, p. 420),

$$min_{\beta,\beta_0}\frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}\xi_i,$$

$$\text{subject to }, \quad y_i(x^T\beta + \beta_0) \geq 1 - \xi_i,\ \xi_i \geq 0\ \forall i$$

The optimal boundary separating binary classes is often nonlinear. While a hyperplane suffices for linear discrimination, complex data requires more flexible decision functions. Kernel methods enable construction of nonlinear separators by implicitly mapping the input space to high dimensions where classes become linearly separable. A kernel defines an inner product between pairs of observations in this transformed space, quantifying similarity. Let x_i and x_i' be two data points and kernel is represented as $K(x, x') = (x \cdot x')$.

3.6.1 Linear Kernel

The linear kernel measures the similarity between two vectors by computing their dot product. For two vectors, x and x' in the original feature space, the linear kernel is given by,

$$K(x_i, x_i') = (x \cdot x'),$$

where p is the number of features. The linear kernel essentially quantifies the similarity between data points based on their correlation in the original feature space. It is suitable for datasets where the classes are separable by a linear boundary.

3.6.2 Radial Kernel

The radial or radial basis function (RBF) kernel measures similarity by considering the Euclidean distance between two data points in the transformed space. It is defined as,

$$K(x_i, x_i') = \exp(-\gamma||x - x'||^2),$$

where γ is a positive constant that influences how far the training point reaches. The radial kernel assigns higher similarity to data points that are closer in the transformed space and rapidly diminishes similarity as points move farther apart. It is effective for capturing non-linear decision boundaries and is suitable for scenarios where data is not linearly separable.

In summary, the linear kernel is suitable for linearly separable data, providing a straightforward measure of similarity based on the dot product. On the other hand, the radial kernel introduces a more flexible approach, capturing complex relationships between data points in a non-linear fashion, making it suitable for a wider range of datasets. The choice between these kernels depends on the specific characteristics of the data and the desired complexity of the SVM model.

3.7 Penalized Logistic Regression

Logistic regression is a common binary classifier that outputs a probability score predicting class membership for a given observation. It works by passing a linear combination of input features through a sigmoid activation to transform real-valued inputs to normalized probabilities between 0 and 1.

Given training data with N samples, each comprising features x and a binary label y, logistic regression models the conditional likelihood $P(y = 1|x)$ - that is, the probability observation x belongs to the positive class $y = 1$. This conditional probability function is denoted as $\pi_{(x)}$, is given by (Liang et al., 2013),

$$P(y_i = 1|x_{ij}) = \pi(x_j) = \frac{e^{x_j'}}{1 + e^{x_j'}}, j = 1, 2, \ldots, p$$

$$f(y_i) = \pi_i^{(y_i)}(1 - \pi_i)^{(1-y_i)}, i = 1, 2, \ldots, N$$

Then the log-likelihood becomes,

$$\ell(\beta, y_i) = \sum_{i=1}^{N} \{y_i \log\pi(x_i) + (1 - y_i)\log(1 - \pi_i)\}$$

Penalized Logistic Regression is an extension of the standard logistic regression that incorporates regularization techniques. Regularization helps prevent overfitting and addresses multicollinearity by introducing penalty terms on the logistic regression model parameters. It is defined as,

$$PLR = \sum_{i=1}^{N} \{y_i \log\pi(x_i) + (1 - y_i)\log(1 - \pi_i)\} + \lambda P(\beta),$$

where λ is a tuning parameter which controls the strength of the penalty, influencing the degree of regularization applied, β is the vector of coefficients and $P(\beta)$ is the penatly term. The choice of penalty type (Lasso, Ridge, or Elastic Net) depends on the specific requirements of the modeling task.

3.7.1 LASSO (L_1 Regularization)

The lasso (least absolute shrinkage and selection operator) combines continuous coefficient shrinkage with automatic variable selection (Tibshirani, 1996). This is achieved through applying an L_1 penalty that encourages sparsity in the fitted model. The L_1 penalty sums the absolute values of the regression coefficients: $P(\beta) = \sum_{j=1}^{p} |\beta_j|$

By incorporating this penalty term, the lasso logistic regression optimization function becomes,

$$PLR = \sum_{i=1}^{N} y_i \log\pi(x_i) + (1 - y_i)\log(1 - \pi(x_i)) + \lambda \sum_{j=1}^{p} |\beta_j|$$

The maximum likelihood solution of the above equation is given by (Liang et al., 2013),

$$\hat{\beta}_{LASSO} = \arg\min_{\beta} \left\{ \ell(\beta, y_i) + \lambda \sum_{i=1}^{N} |\beta_j| \right\}$$

Through imposing both shrinkage and variable selection, the lasso provides a method to remove irrelevant features. Tuning λ navigates this sparsity-accuracy tradeoff.

While LASSO finds extensive use in various applications, it does have some limitations. One notable drawback is its lack of robustness when dealing with highly correlated predictor variables. In such cases, LASSO tends to arbitrarily select one variable and disregard the others.

3.7.2 Ridge (L_2 Regularization)

Ridge regression incorporates an L_2 penalty to address multicollinearity among explanatory variables (Hoerl and Kennard., 1970). The L_2 penalty term sums the squared values of all coefficients β_j^2:

$$P(\beta) = \sum_{j=1}^{p} \beta_j^2$$

This leads to the penalized logistic regression loss function with ridge penalty:

$$PLR = \sum_{i=1}^{N} \{y_i \log\pi(x_i) + (1 - y_i)\log(1 - \pi_i)\} + \lambda \sum_{i=1}^{N} \beta_j^2$$

The maximum likelihood solution of the above equation is given by (Liang et al., 2013),

$$\hat{\beta}_{Ridge} = \arg\min_{\beta} \left\{ \ell(\beta, y_i) + \lambda \sum_{i=1}^{N} \beta_j^2 \right\}$$

Here, the tuning parameter λ governs the degree of shrinkage applied to the coefficients of explanatory variables but does not force any of them to be precisely zero. Consequently, ridge regression does not yield a sparsity model. Despite lacking sparsity, ridge regression is favored in high-dimensional data scenarios due to the expectation of significant correlation among explanatory variables.

3.7.3 Elastic Net

The elastic net was proposed to address limitations of the LASSO by combining both L_1 and L_2 regularization penalties. The elastic net logistic regression function is defined as (Zou and Hastie, 2005):

$$PLR = \sum_{i=1}^{N} \{y_i \log\pi(x_i) + (1 - y_i)\log(1 - \pi_i)\} + \lambda_1 \sum_{i=1}^{N} |\beta_j| + \lambda_2 \sum_{i=1}^{N} \beta_j^2$$

This formulation leverages the sparsity-inducing L_1 penalty from lasso along with the L_2 stabilization penalty from ridge regression. The two non-negative hyperparemeters λ_1 and λ_2 control the relative weighting. By integrating both shrinkage and variable selection, the elastic net handles issues of overfitting, multicollinearity, and selection instability faced by standard logistic regression. The maximum likelihood elastic net solution is obtained by:

$$\hat{\beta}_{Elastic} = \arg\min_{\beta} \left\{ \ell(\beta, y_i) + \lambda_1 \sum_{i=1}^{N} |\beta_j| + \lambda_2 \sum_{i=1}^{N} \beta_j^2 \right\}$$

Tuning λ_1 and λ_2 navigates the accuracy-sparsity tradeoff to find a stable, predictive feature subset even in high dimensions with correlation present.

3.8 Filter Methods

Handling high-dimensional datasets has become increasingly important in machine learning. High-dimensional data presents modeling challenges including overfitting, noise accumulation, and exponential search spaces (curse of dimensionality, (Bommert et al., 2020)). Hence, feature selection plays a vital role in enabling efficient data analysis and machine learning.

Filter methods provide a way to improve algorithm performance while reducing runtime overhead. These methods assess each feature by assigning scores to them and rank the features based on their respective scores. Consider a dataset with N instances across p features denoted as, $X_1, X_2, \ldots, X_p$ and Y be the target variable representing classes. The following sections provide an overview of different filter methods.

3.8.1 Variance Filter

The variance filter computes a score for each feature X_k based on its variance across samples. The score (J_{variance}) for each feature X_k is calculated as follows (Bommert et al., 2020):

$$J_{\text{variance}}(X_k) = \frac{1}{N-1} \sum_{i=1}^{N} (x_i^{(k)} - \bar{x^{(k)}})^2$$

Where $x_i^{(k)}$ denotes the observed value for feature X_k in sample i and $\bar{x^{(k)}}$ is the mean value of X_k across the N samples.

Essentially, features exhibiting a wider spread or higher deviation from their average value across samples obtain higher variance scores. The filter targets identifying and removing variables dominated by noise with little fluctuation in their measurements across observations. Features with minimal variance provide negligible explanatory or modeling value. Their removal concentrates signal retained in highly variant features.

Notably, this filter best applies to raw data leaving features on their native scales prior to standardization. Standardizing enforces equal unit variance across inputs inherently rather than assessing intrinsic variability indicative of information content. By avoiding this preprocessing step, the variance filter can effectively screen raw inputs on their original scale of fluctuation.

3.8.2 JMIM Filter

Entropy ($H(Y)$) quantifies the uncertainty of a random variable Y by measuring the amount of information needed to describe its value. Conditional entropy ($H(Y|X)$) determines the remaining entropy of Y after observing X. The mutual information ($I(Y;X)$) between Y and X is the reduction in Y's entropy given the information about X which is represented as,

$$I(Y;X) = H(Y) - H(Y|X)$$

Let S denote a set of already selected features and $I(Y;X_k,X_j)$ represent the amount of information about the target variable Y that is jointly provided by the features X_k and X_j, where X_j belongs to already selected features S.

When selecting features, the goal is to extract variables X_j that maximize $I(Y;Xj)$ to be highly informative of Y. However, selecting only based on that criterion may result in redundant features providing similar information about Y.

To promote diversity, the Joint Mutual Information Maximization (JMIM) criterion scores each candidate feature X_k based on (Bommert et al., 2020),

$$J_{MIM}(X_k) = \min_{X_j \in S} \{I(Y;X_k,X_j)\}$$

JMIM finds the minimum mutual information between X_k and any current feature $X_j \in S$. This measures the unique complementary information X_k can provide about Y beyond the existing selections.

3.8.3 Permutation Filter

Permutation filter assesses feature significance in a random forest by examining out-of-bag (oob) instances for each tree, representing the instances which were not used during that particular tree's training. For a specific feature (X_k), the oob instances are classified, and the feature is randomly permuted. The drop in classification accuracy from original oob instances to permuted ones determines the permutation importance score ($J_{permutation}$) (Bommert et al., 2020). A crucial feature will exhibit a notable accuracy decrease when permuted.

$$\begin{aligned} J_{\text{permutation}}(X_k) = &\text{ accuracy for original oob instances} \\ &- \text{accuracy for oob instances with permuted values of } X_k \end{aligned}$$

3.8.4 Impurity Filter

In a random forest tree, a node N represents a subset of training samples that reach a particular branch based on splits along the tree. Node purity refers to samples belonging to the same class (pure) versus multiple classes (impure).

Let $N^{(k)}$ indicate nodes where a split occurs based on candidate feature X_k. The Gini index quantifies impurity of nodes before and after splitting on X_k.

The impurity filter scores feature X_k using mean decrease in impurity caused by splits on that feature across all involved nodes $N^{(k)}$ (Bommert et al., 2020):

$$J_{\text{impurity}}(X_k) = \frac{\sum_{i \in N^{(k)}} (\text{impurity before node } i - \text{impurity after node } i)}{|N^{(k)}|}$$

Features yielding higher impurity decreases through tree splits are considered more important for reducing uncertainty of the prediction task. This is aggregated over trees in the ensemble model.

3.9 Hyperparameter Optimization

Hyperparameter optimization refers to the automated process of discovering the optimal set of hyperparameters for a machine learning model, aiming to maximize the performance of a specific problem.

Hyperparameters are knobs that govern how the model trains and makes predictions - like number of trees in a random forest or regularization strength in logistic regression. Unlike model parameters, hyperparameters are set before training begins rather than learned through the optimization process.

Finding optimal hyperparameter values is crucial since they greatly impact model accuracy and generalization. However, going through all possible values exhaustively is error-prone and expensive (Bischl et al., 2021). Hyperparameter optimization uses specialized algorithms to strategically and efficiently explore combinations to identify prime hyperparameter values.

Some common hyperparameter optimization methods include (Bischl et al., 2021):

- Grid search - Evaluates hyperparameters from all set combinations
- Random search - Tests random values from predefined ranges
- Bayesian optimization - Uses a probabilistic model to guide sampling of promising values

By automatically tuning hyperparameter configurations, more effective machine learning pipelines can be constructed with less manual effort and guesswork. This enables reproducibility maximizing model performance.

Tables 1 through 3 (Bischl et al., 2021) outline recommended hyperparameter tuning ranges for key parameters across the classification methods.

Certain hyperparameter values are optimized on a logarithmic scale rather than linear scale. Transforming proposed values logarithmically enables higher resolution tuning specificity for smaller magnitudes versus larger magnitudes. The "Trafo" column in the tables indicates whether the hyperparameter values on a logarithmic scale before being passed to the learning algorithm.

3.9.1 Random Forest

Random forests involve tuning a large number of hyperparameters that govern both the individual decision trees as well as the overall forest structure. Tree-level hyperparameters control aspects like tree depth and splitting criteria. Forest-level hyperparameters determine properties like number of trees, sampling strategy for building trees, and the randomness applied during training such as number of variable candidates considered at each split. Table 1 depicts the hyperparameter ranges of random forest.

Hyperparameter	**Range**	**Description**
$mtry$	1 to p	Number of feature considered for splitting at each node.
$num.trees$	1 to 2000	Number of decision trees composing the ensemble.
$sample.fraction$	[0.1, 1]	Proportion of training samples drawn randomly.

Table 1: Random Forest Hyperparameter Ranges

3.9.2 Support Vector Machines

The goal of a support vector machine (SVM) is to construct a maximal margin hyperplane between classes in the feature space. The key tuning hyperparameters include the regularization penalty, kernel choice and associated kernel parameters. Table 2 summarizes the hyperparameter ranges of SVM.

SVMs leverage transformed feature spaces to find maximal margin linear separators between classes formulated through regularization objectives. Performance depends heavily on selecting the right kernel and controlling model complexity.

3.9.3 Penalized logistic regression

Penalization is controlled through a regularization hyperparameter (λ or s) that governs the degree variables are constrained - either via L1 norms to induce sparsity (the "lasso")

Hyperparameter	Range	Trafo	Description
cost	[-12, 12]	2^x	Margin violations costs.
kernel	radial, linear	-	Type of kernel function.
gamma	[-12, 12]	2^x	Controls shape of kernel function

Table 2: Support Vector Machine Hyperparameter Ranges

or more commonly L2 norms to shrink coefficients while retaining all inputs (ridge regression). The range of hyperparameter values are presented in table 3.

Optimizing these regularization hyperparameters is crucial for balancing model complexity and avoiding overfit. The sweet spot enables compressing coefficients down from extreme values without over-shrinking important discriminative signals for classification. Grid or random search is computationally practical for tuning penalized logistic regression.

Hyperparameter	Range	Trafo	Description
s	[-12, 12]	2^x	The regularization strength that controls extent of coefficient shrinkage towards 0.
alpha	[0, 1]	-	Controls the balance between L1 and L2 regularization.

Table 3: Penalized logistic regression Hyperparameter Ranges

3.10 Performance Measures

Performance measures are metrics used to assess the effectiveness of a machine learning model in accurately categorizing instances into predefined classes. These measures provide insights into how well the model is performing and are crucial for evaluating its reliability.

3.10.1 Confusion matrix

In the context of classification models, a confusion matrix is an evaluation tool that provides a detailed breakdown of the model's predictions and actual outcomes across different classes. In a binary classification scenario, the confusion matrix consists of four key elements:

1. True Positive (TP): This represents the instances that were correctly predicted as positive by the model.
2. True Negative (TN): These are instances correctly predicted as negative by the model.
3. False Positive (FP): Also known as a Type I error, FP represents instances predicted as positive by the model but were, in fact, negative.
4. False Negative (FN): Termed as a Type II error, FN denotes instances predicted as negative by the model but were actually positive.

The confusion matrix is a square table, where the rows represent the true classes, and the columns depict the predicted classes as shown in 4. The primary diagonal of the matrix contains the counts of instances that were correctly classified, offering insights into the model's accurate predictions. Conversely, the off-diagonal elements highlight instances of misclassification by the model. (Han et al., 2012, p. 264)

		Predicted	
		Positive	Negative
Actual	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 4: Confusion Matrix for Binary Classification

3.10.2 Accuracy

Accuracy represents the overall correctness of a model by measuring the proportion of correctly classified instances out of the total instances. The accuracy is calculated using the formula:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

In the context of a confusion matrix, where TP is True Positive, TN is True Negative, FP is False Positive, and FN is False Negative, accuracy can be expressed as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Accuracy provides a general overview of a model's performance, indicating the percentage of correct predictions. (Han et al., 2012, p. 266)

3.10.3 AUC Score

The area under the receiver operating characteristic curve (AUC) provides key evaluation measure for classification models. The ROC curve plots the true positive rate against the false positive rate across classification thresholds. AUC measures the entire two-dimensional area underneath this curve, representing the model's cumulative ability to correctly rank positive samples higher than negative samples irrespective of any single threshold (Kotu and Deshpande, 2015, p. 260). An AUC of 0.5 signifies random predictive ability, while a score of 1 indicates perfect classification.

4 Statistical analysis

This section performs statistical analysis on the dataset using using the statistical methods outlined in Section 3.

4.1 Pre-processing

Prior to conducting gene expression analysis, a pre-filtering step is implemented on both Standard diet and Western diet data. This involves the elimination of genes with fewer read counts, aiding in the reduction of memory size and enhancing the speed of count modeling in DESeq2. The pre-filtering criterion is set to retain only those genes that have a count of at least 10 in a minimal number of samples. Out of the initial 35,727 genes, a total of 14,994 genes were excluded from further analysis due to having counts below the specified threshold of 10 in the samples.

Variance Stabilizing transformation (VST) is performed using *vst()* function from *DESeq2* package. VST eliminates the relationship between variance and mean, specifically addressing the issue of high variance of the logarithm of count data when the mean is low. In RNA-seq count data, genes with higher counts tend to exhibit higher variance across samples. This heteroscedasticity (non-constant variance) can violate assumptions made by many statistical tests. VST transforms the count data so that the variance becomes stabilized across different mean expression levels. Stabilizing the variance is useful for visualizations and statistical analysis to compare gene expression levels, detect differential expression, and perform classifications.

4.2 Descriptive analysis

4.2.1 Differential Expression Analysis

The DESeq() function performs differential expression analysis and returns a DESeqDataSet containing the results. The results() method extracts a table with statistics for comparing expression between conditions, such as week 3 versus week 30 depicted in Figure 1.

The key output columns are:

- **baseMean**: Average of normalized count values across all samples.

- **log2FoldChange**: Effect size estimate indicating degree of upregulation (positive value) or downregulation (negative value) between conditions. For the gene ENSMUSG00000000056, the log2FoldChange value of 0.7807839 indicates that expression of this gene exhibits a 0.7807839 fold increase, equivalent to a $2^{0.7807839} = 1.72$ fold upregulation at week 30 compared to week 3.
- **lfcSE**: Standard error of the log2FoldChange estimate.
- **p-value**: For each gene, DESeq2 conducts a hypothesis test, and the outcome is expressed as a p value. It provides the statistical evidence against the null hypothesis. Lower p-values indicate less probability that the observed change was random.
- **padj**: DESeq2 employs the Benjamini-Hochberg (BH) adjustment method, yielding adjusted p values (padj). These adjusted p values account for the fraction of false positives (false discovery rate, FDR) among genes deemed significant based on the associated p-value threshold. The BH-adjusted p values provide a reliable measure for identifying differentially expressed genes while controlling false positives.

  For example, an adjusted p-value below a threshold like 0.05 indicates that if all genes with p-values lower than that were called significant, no more than 5% would be false positives.

```
log2 fold change (MLE): weeks 3 vs 30
Wald test p-value: weeks 3 vs 30
DataFrame with 20733 rows and 6 columns
                     baseMean log2FoldChange     lfcSE       stat      pvalue       padj
                    <numeric>      <numeric> <numeric>  <numeric>   <numeric>  <numeric>
ENSMUSG00000000001  2337.43289      0.0786212 0.0894105  0.8793284 3.79223e-01 0.52297597
ENSMUSG00000000028    53.34408     -0.2292930 0.1558897 -1.4708671 1.41327e-01 0.24708972
ENSMUSG00000000037     1.54868      0.0325518 0.7803828  0.0417126 9.66728e-01 0.98155089
ENSMUSG00000000049 56875.24366      0.1959676 0.1497410  1.3087100 1.90633e-01 0.31130938
ENSMUSG00000000056  1538.11880      0.7807839 0.1577999  4.9479372 7.50041e-07 0.00001197
...                       ...            ...       ...        ...         ...        ...
ENSMUSG00000118450    0.497050       2.561637  1.060488   2.415527   0.0157124   0.041951
ENSMUSG00000118458    1.239530       0.000000  5.642633   0.000000   1.0000000   1.000000
ENSMUSG00000118461    1.533480      -0.926232  0.666083  -1.390565   0.1643574   0.277888
ENSMUSG00000118471    0.625387      -1.947196  5.628331  -0.345963   0.7293703   0.821955
ENSMUSG00000118487    0.863963      -0.294869  0.842664  -0.349925   0.7263953   0.819779
```

Figure 1: Result of weeks 3 vs 30 Differential Expression Analysis

4.2.2 Principal Components Analysis

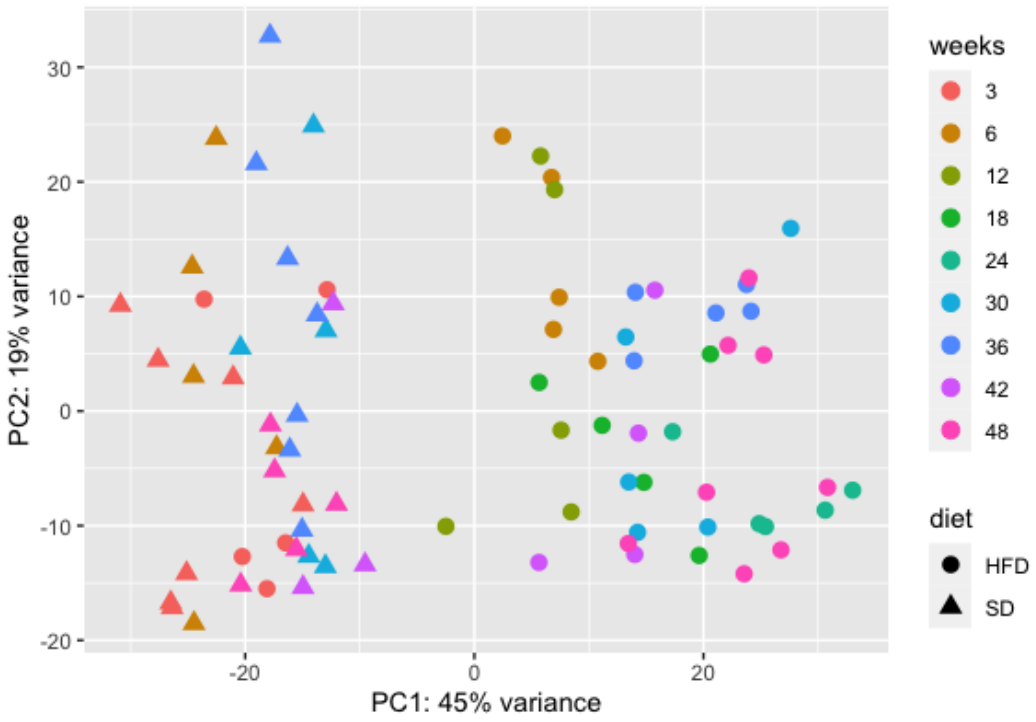


Figure 2: PCA Plot

Figure 1, depicts PCA plot of both Standard diet and Western Diet (or High Fat Diet-HFD) data from week 3 to week 48. The first principal component (PC1), displayed on the x-axis, accounts for 45% of the total variance in the gene expression data. PC1 stratifies the data based on diet, differentiating between the standard diet and western diet groups. Samples from mice fed with two different diets are separated along this axis, indicating substantial differences between these conditions.

The second principal component (PC2) on the y-axis captures 19% of the variance. In the early time points, the western diet samples at week 3 overlap with the standard diet samples, indicating minimal differential expression at baseline. The week 6 and 12 western diet samples remain closer to the standard diet samples with some intermediate displacement, signifying more modest differences developing at these early stages.

However, the HFD samples show vertical separation spanning week 6 to week 48 along the PC2 axis. This reflects substantial variance between HFD mice arising later in the time course progression which means that the samples from closer weeks exhibits more

similar expression profiles compared to those farther apart in timing. Together PC1 and PC2 captures a total variance of 64%.

4.2.3 Univariate Analysis

The predictive capacity of each gene was assessed independently by calculating the area under the ROC curve (AUC) scores. It quantifies how accurately a binary classifier distinguishes outcomes, with an AUC of 0.5 representing random chance and 1 indicating perfect classification.

The observed gene expression data contains 22 samples, with binary tumor development labels. The AUC score is calculated independently for each of the 20,531 genes, measuring its capability to accurately classify the tumor status of the mice. Figure 8 in the Appendix on page 51, shows the distribution of AUC scores with actual labels and the distribution of AUC scores with randomly permuted labels.

The distribution of AUC scores for individual genes based on the true labels, shown in Figure 8(a), was concentrated in the 0.5-0.6 range. This distribution aligns with what would be expected for classifiers operating at random chance levels for a binary prediction task. To confirm that the genes did not have meaningful univariate predictive power, AUC scores were also calculated after randomly shuffling the tumor status labels (Figure 8(b)). The permutation-based AUC distribution closely mirrors the pattern seen using the true labels, with a slight rightward shift showing a higher density of genes with AUCs centered on 0.55. The similarity between the actual and permuted AUC score distributions suggests that genes, as considered in this analysis, do not have strong predictive power for the given tumor development labels.

4.3 Feature Selection

In this section, feature selection is applied on the gene expression data to identify subsets of genes that can be beneficial to predict tumor development. Selecting the most informative genes allows for accurate tumor development prediction while reducing overfitting.

4.3.1 Significant down-regulated genes

Significant down regulated genes are identified through differential expression analysis. The comparisons are made between gene expression levels at week 3 versus later timepoints - specifically week 6, 30, 36, 42, and 48. Genes are then filtered based on a threshold of log2foldChange less than or equal to -1 and a False Discovery Rate (FDR) adjusted p-value less than 0.05 to select significantly downregulated genes. This threshold is set based on biological reasoning.

A Venn diagram (Figure 3) was then employed to assess the overlap of down-regulated genes between week 3 vs week 6 and the collective set of week 3 vs 30, 36, 42, and 48. The results indicated that 102 genes were down-regulated at week 3 compared to week 6, 968 genes displayed significant changes in expression across weeks 30, 36, 42, and 48, and the intersection of these two sets comprised 97 genes.

The downregulated genes uniquely identified in the week 30+ timepoint comparisons amounted to 871 genes. These genes exhibited significant differential expression in comparisons to baseline at week 3, especially emerging at the later 30+ week timepoints. This subset is taken forward for classification analysis.

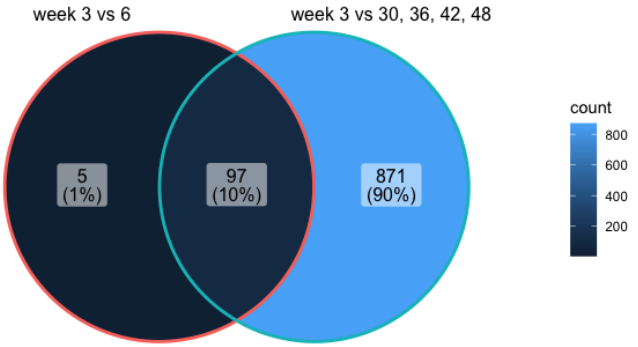


Figure 3: Significant down regulated genes

4.3.2 Significant up-regulated genes

Significant up regulated genes are identified through differential expression analysis. Differentially expressed genes were assessed through pairwise comparisons, such as week 3 vs week 6, week 3 vs week 30, week 3 vs week 36, week 3 vs week 42, and week 3 vs week 48. Up-regulated genes are filtered based on having a log2fold change greater

than or equal to 1 and an adjusted p-value less than 0.05. This threshold is set based on biological reasoning.

A Venn diagram was then employed to assess the overlap of up-regulated genes between week 3 vs week 6 and the collective set of week 3 vs 30, 36, 42, and 48. Figure 4 shows the overlap between the upregulated genes found in the week 3 versus 6 comparison and those identified across the later 30+ week timepoints. 125 genes were down-regulated at week 3 compared to week 6 and 920 genes displayed significant changes in expression across weeks 30, 36, 42, and 48. There were 76 genes in common between these two groups. Therefore, the total count of uniquely significant up-regulated genes across all time points was 844. This analysis provides insights into the changes in gene expression over time, highlighting 844 genes that exhibit significant up-regulation when compared to week 6.

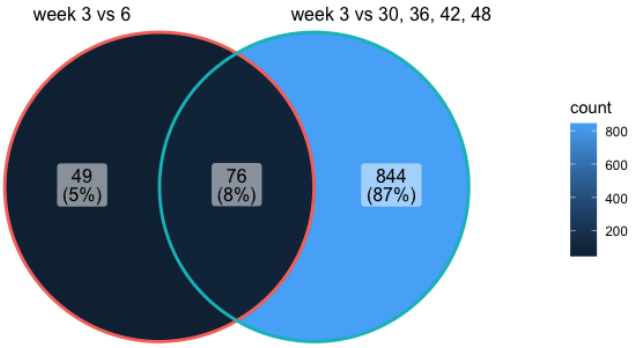


Figure 4: Significant up regulated genes

4.3.3 Model down-regulated genes

A 4-parameter log-logistic model was utilized to identify genes with significant downward trajectories in expression over the 48 week time course experiment. Utilizing the drm() function, the 4PLL model was fitted to model the gene expression count as a function of time in weeks. The determination of downregulation was based on the absolute effective dose (ED) threshold, indicating the timepoint where the gene expression curve decreases by log2(1.5) from its initial value between week 3 and week 48.

Figure 5 illustrates a concentration-response cure for a specific gene (ENSMUSG000000000056). Here, x-axis represents time-points in units of weeks and y-axis represents the gene expression counts. The horizontal dotted line signifies the downregulation threshold level,

and the intersection point depicts the time at which the gene's expression decreases by log2(1.5) from its initial value.

Among all the genes subjected to the 4PLL model fits, 500 genes fulfilled these criteria, showcasing a significant downward expression trajectory. This approach selects genes that demonstrate substantial downregulation over time.

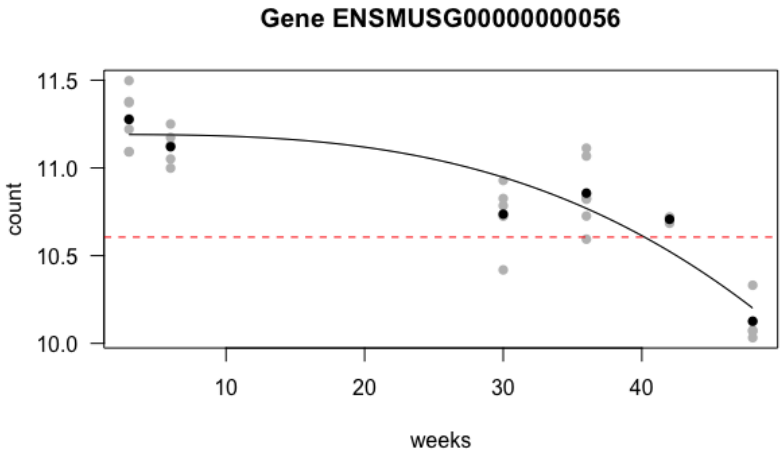


Figure 5: Model down regulated genes

4.3.4 Model up-regulated genes

To identify upregulated genes with relevant expression patterns over time, a 4-parameter log-logistic (4pLL) model was fit to each gene's expression profile across the time course from week 3 to 48.

For each gene, the 4pLL model was fit using the drm() function, modelling expression count over time in weeks. The fitted model object contains parameter estimates customizing the curve to that gene's specific expression pattern. Using the absolute effective dose (ED) calculation, genes were selected where the gene expression profile rose by at least log2(1.5) between timepoints within the experimental range from week 3 to 48. This is visualized in Figure 6, a concentration-response cure for a gene ENSMUSG000000000552. Here, x-axis represents time-points in units of weeks and y-axis represents the gene expression counts. The horizontal dotted line represents the upregulation threshold level. The point where the curve meets the horizontal line represents the time point at which gene's expression increases by log2(1.5) from its initial value.

Out of all the 4PLL model fits, 362 genes met these criteria of showing a significant upwards expression trajectory during the experiment. This approach selects genes that demonstrate substantial upregulation over time.

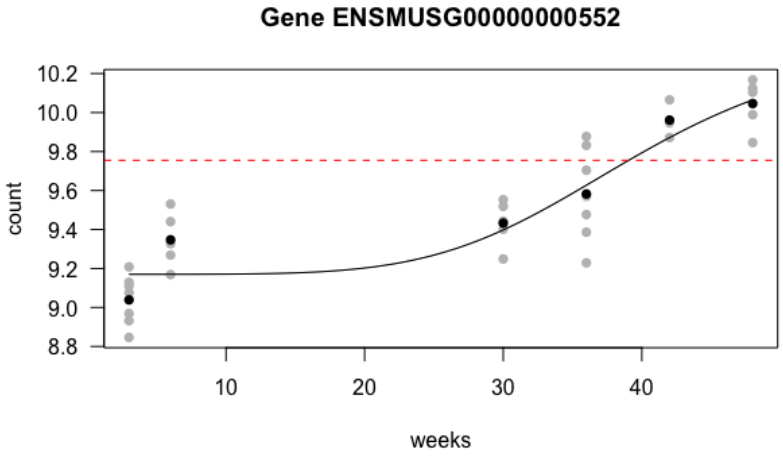


Figure 6: Model up regulated genes

4.3.5 Naive genes

For the identification of genes with variability, a straightforward approach was employed by selecting the top 500 genes exhibiting the highest variability. This method, termed "Naive genes" involves considering the genes that demonstrate the greatest degree of gene expression variation, regardless of specific temporal patterns or regulatory responses. By focusing on the genes with high variability, this approach captures a diverse set of candidates, potentially encompassing genes responsive to various factors or exhibiting behaviors in their expression levels across different experimental contexts.

4.4 Classification Methods

To predict tumor development in mice, four classification algorithms were employed: Random Forest (or Ranger), SVM with linear kernel, SVM with radial kernel, and Penalized Logistic Regression (Penalized LR).

Stratification was performed prior to modeling inorder to ensure balanced class distribution. Samples were split into k=3 folds while preserving the ratio of cases to controls in each fold.

Hyperparameter optimization was implemented to select the best model configuration for four classification algorithms. Hyperparameter values presented in tables 1-3 are employed for this purpose. Grid search was utilized for hyperparameter tuning.

Nested cross-validation was employed, using an inner 3-fold cross-validation loop for hyperparameter selection and outer 3-fold cross-validation loop for performance evaluation. This nested CV approach minimizes overfitting.

Each experiment was run for five iterations with accuracy and confusion matrix as performance measures. Accuracy scores and confusion matrices are averaged over the five iterations. The results are presented in tables 5 to 8 and are discussed in the subsequent sections.

4.4.1 Random Forest

Table 5 summarizes the performance of a Random Forest classifier applied to various gene sets for predicting tumor, as quantified by confusion matrices and accuracy scores.

For model up-regulated genes, prediction accuracy was 52.2%. The confusion matrix shows that among 10 actual positive samples, the model correctly classified 5 as tumor-bearing (true positives). Among the 12 actual negative samples, it correctly classified 6.6 as non-tumor, while misclassifying 5.4 samples.

The accuracy was similar at 49.8% for model down-regulated genes. Here out of 10 positive samples, 5.4 were correctly detected, while 4.6 were missed. For the 12 negative class samples, prediction success was also more modest at 5.6 correct vs. 6.4 false positives.

Naive gene set had a low accuracy of 48.4%, with greater imbalance between precision and recall. Significant upregulated genes achieved 51% accuracy, with 4.6 true positives out of 10, while significant downregulated genes attained 47.6% accuracy with higher false negatives.

In summary, the Random Forest algorithm exhibits varied performance across different gene categories.

Genes	Confusion Matrix	Accuracy
Model up	$\begin{bmatrix} 5 & 5.4 \\ 5 & 6.6 \end{bmatrix}$	52.2%
Model down	$\begin{bmatrix} 5.4 & 6.4 \\ 4.6 & 5.6 \end{bmatrix}$	49.8%
Naive	$\begin{bmatrix} 4.8 & 6.2 \\ 5.2 & 5.8 \end{bmatrix}$	48.4%
Significant up	$\begin{bmatrix} 4.6 & 5 \\ 5.4 & 7 \end{bmatrix}$	51%
Significant down	$\begin{bmatrix} 2.8 & 4.2 \\ 7.2 & 7.8 \end{bmatrix}$	47.6%

Table 5: Confusion Matrix and Accuracy of Random Forest

4.4.2 SVM Linear

The SVM Linear model demonstrates moderate performance in predicting tumor status across the different gene sets, with accuracy ranging from 50-62%. This is summarized in Table 6.

For model up-regulated genes, the SVM Linear model achieves 50.16% accuracy. As shown in the confusion matrix, out of 10 actual positive samples, only 1.8 are correctly predicted as tumor (true positives). Out of the 12 actual negative samples, 9.3 are correctly predicted as non-tumor (true negatives). Performance is slightly better for model down-regulated genes at 53.5% accuracy. Here, 3.4 out of 10 actual positive samples are correctly classified as tumor-positive (true positives). For the 12 actual negative samples, 8.4 are correctly predicted as non-tumor (true negatives).

The naive gene set enables a better performance for SVM Linear at 61.33% comparatively. Sensitivity is higher for both classes, detecting 5 out of 10 positives correctly and 8.4 out of 12 negatives. Accuracy dips for significant upregulated genes (52%) with greater class imbalance favoring true negatives. Almost 10 out 12 samples were correctly predicted as true negatives. Lastly, significant downregulated genes attain strong 61.83% accuracy with low false negatives.

Overall SVM Linear shows promise in the 50-60% range but also difficulties in correct predictions. Compared to other evaluated models, SVM Linear performance is middling with some peaks but also clear pitfalls.

Genes	Confusion Matrix	Accuracy
Model up	$\begin{bmatrix} 1.8 & 2.8 \\ 8.3 & 9.3 \end{bmatrix}$	50.16%
Model down	$\begin{bmatrix} 3.4 & 3.6 \\ 6.6 & 8.4 \end{bmatrix}$	53.5%
Naive	$\begin{bmatrix} 5 & 3.6 \\ 5 & 8.4 \end{bmatrix}$	61.33%
Significant up	$\begin{bmatrix} 1.2 & 1.8 \\ 8.8 & 10.2 \end{bmatrix}$	52%
Significant down	$\begin{bmatrix} 5.4 & 3.8 \\ 4.6 & 8.2 \end{bmatrix}$	61.83%

Table 6: Confusion Matrix and Accuracy of SVM Linear

4.4.3 SVM Radial

Table 7 presents the confusion matrices and accuracy scores for different gene models generated by the SVM Radial method. The SVM Radial model demonstrates accuracy in the range of 49-57% across the gene sets.

For model up-regulated genes, it achieves 56.8% accuracy. The confusion matrix shows 2.8 out of 10 true positives are detected, while 9.6 out of 12 true negatives are correctly predicted. Performance drops slightly for model down-regulated genes to 54.67%, with 4.8 achieved true positives and 7.2 true negatives. Naive genes exhibit a similar 54% accuracy. Here, the model predicts 9 true negatives and a very low true negatives of 2.8.

Accuracy for significant upregulated genes sees the lowest value of 49.83%. The model struggles with more in predicting true postives correctly. 8.4 true negatives are correctly predicted. Lastly, significant downregulated genes attain 50.66% accuracy. For this category, 3 out of 10 true positives and 8.2 out of 12 true negatives are identified.

4.4.4 Penalized Logistic Regression

Table 8 presents the confusion matrices and accuracy scores for different gene models generated by the Penalized Logistic Regression method. The model demonstrates moderately successful prediction of tumor status across categories, achieving accuracy between 46-57%.

Genes	Confusion Matrix	Accuracy
Model up	$\begin{bmatrix} 2.8 & 2.4 \\ 7.2 & 9.6 \end{bmatrix}$	56.8%
Model down	$\begin{bmatrix} 4.8 & 4.8 \\ 5.2 & 7.2 \end{bmatrix}$	54.67%
Naive	$\begin{bmatrix} 2.8 & 3 \\ 7.2 & 9 \end{bmatrix}$	54%
Significant up	$\begin{bmatrix} 2.6 & 3.6 \\ 7.4 & 8.4 \end{bmatrix}$	49.83%
Significant down	$\begin{bmatrix} 3 & 3.8 \\ 7 & 8.2 \end{bmatrix}$	50.66%

Table 7: Confusion Matrix and Accuracy of SVM Radial

For model up-regulated genes, accuracy is highest at 57.16% when compared to other datasets. The confusion matrix indicates depicts 3.2 of 10 true positives and 9.4 of 12 true negatives correctly identified. Performance drops to 49.67% on model down-regulated genes. Fewer true positives (3) are attained alongside more incorrectly predicted negatives (4.3).

Naive genes enable 55.83% accuracy, with 4 of 10 true positives and 8.2 of 12 true negatives. Significant upregulated genes prove most difficult, with lower true positives (2.4) leading to just 46.33% overall accuracy. Also, 7.8 of 12 are true negatives. Lastly, significant downregulated genes achieve intermediate 54.67% accuracy with equal true positives and false positives (3.8), and 8.6 true negatives.

Genes	Confusion Matrix	Accuracy
Model up	$\begin{bmatrix} 3.2 & 2.6 \\ 6.8 & 9.4 \end{bmatrix}$	57.16%
Model down	$\begin{bmatrix} 3 & 4.3 \\ 7 & 7.8 \end{bmatrix}$	49.67%
Naive	$\begin{bmatrix} 4 & 3.8 \\ 6 & 8.2 \end{bmatrix}$	55.83%
Significant up	$\begin{bmatrix} 2.4 & 4.2 \\ 7.6 & 7.8 \end{bmatrix}$	46.33%
Significant down	$\begin{bmatrix} 3.4 & 3.4 \\ 6.6 & 8.6 \end{bmatrix}$	54.67%

Table 8: Confusion Matrix and Accuracy of Penalized Logistic Regression

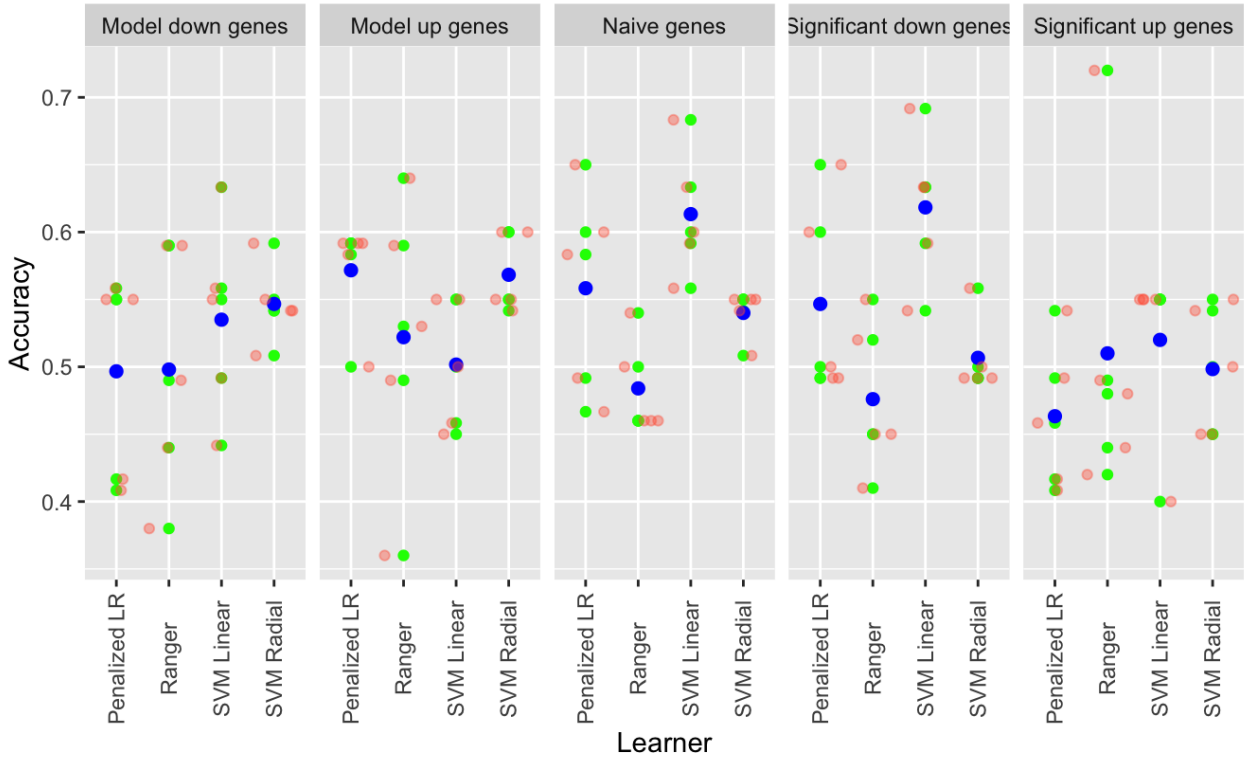


Figure 7: Accuracy by Learner and Dataset

Figure 7 provides a comparative visualization of mean classification accuracy across classification methods (or learners) and different gene sets, compiled from five experimental

iterations. Each green dot signifies one of the five accuracy values obtained per model per gene set. To enable visualization of highly overlapping points, red jitter markers are displayed representing slight modifications to the green data points. Notably, the blue dots denote the mean accuracy of all five experiments.

For model down-regulated genes, the overall accuracy scores vary between 38% and 64%, with average accuracy scores ranging from 50% and 55%. SVM radial outperforms other methods with an average score of 55%. This is followed by SVM linear with 54% average score. Penalized LR and random forest exhibit comparable performance, both achieving around 50% accuracy.

The performance of classification methods on model up-regulated genes is slightly better than for model down-regulated genes. Here, the mean accuracy scores are between 50% and 57%, while the overall accuracy score span from 36% to 64%. Penalized LR and SVM radial demonstrate similar performance, achieving accuracy scores of 57-58% accuracy scores. This is trailed by random forest with 52% average accuracy score, while SVM linear shows the least performance with 50% mean accuracy.

Naive genes dataset displays varied performance, with average accuracy score varying between 48% and 62%, and overall accuracy scores are in the range 46% to 68%. SVM linear shows the best performance among all with 62% mean accuracy score. Penalized LR and SVM radial achieve mean accuracy scores of 56% and 54% respectively, while random forest exhibits the lowest performance with 48% mean accuracy score.

Significant down-regulated genes exhibit similar performance to naive genes. Here, the overall accuracy scores range between 41% and 69%, with mean accuracy scores varying from 47% to 62%. SVM linear demostrates the best performance among all with 63% mean accuracy score. Penalized LR and SVM radial achieve mean accuracy scores of 55% and 51% respectively. Random forest exhibits lowest performance with 47% mean accuracy score.

Significant up-regulated genes demonstrates slightly lower performance compared other gene sets. Overall accuracy scores range between 40% and 73%. The mean accuracy scores are in the range of 46% to 52%. SVM linear shows a comparatively better performance with 52% mean accuracy which is followed by random forest and SVM radial with 50-51% mean accuracy. Penalized LR shows the lowest performance with 46% mean accuracy score.

In summary, the overall average accuracy ranges substantially from 46-62%, however the mean is only around 52%. This indicates tumor development prediction from gene data is generally mediocre. No model significantly outperforms others consistently. Random Forest, SVM (Linear & Radial), and Penalized Logistic Regression all exhibit similar promise and pitfalls (mean accuracy scores barely vary). Certain gene sets like significant upregulated give a slight accuracy boost (peaking 50-62%) versus lowest performance for significant upregulated genes (46-53%). Peak accuracy scores around 60% are moderately successful but still far from precise.

4.5 Filter Methods

Given the moderate performance across the panel of classification algorithms, additional filter methods are implemented to improve modeling accuracy. Four filter approaches are implemented: variance, impurity score, joint mutual information maximization, and the permutation filter. These techniques are applied on the previously selected gene sets discussed as in Section 4.3 and also on all genes, without any prior feature selection.

The subsequent sections present the filtering approaches and comparisons of predictive accuracy.

4.5.1 Variance filter

Figure 9 in Appendix on page 52, illustrates the mean classification accuracy for gene sets filtered by variance across different classification methods (or Learners), averaged over five experiments.

The results indicate that all genes set consistently performs well across all classification methods, exhibiting mean accuracy scores ranging from 68% to 85%. Overall accuracy scores vary between 66% and 93%. Notably, random forest outperforms other methods with an average accuracy score of 85%, followed by SVM linear with approximately 77% accuracy. SVM radial achieves an average accuracy score of 75%, while Penalized LR performs relatively less favorably with a mean accuracy score of 68%.

For the model up-regulated genes set, moderate performance is observed, with average accuracy scores ranging from 50% to 60%. Overall accuracy scores span from 42% to 73%. Penalized LR demonstrates slightly better performance with an average accuracy

score of 60%, followed by SVM linear at 59%. Random forest and SVM radial exhibit similar performance with average accuracy scores around 50%.

Similarly, the model down-regulated genes dataset performs comparably to the model up-regulated genes dataset, with overall accuracy ranging from 42% to 69%. Random forest outperforms other methods with a mean accuracy score of 59%. This is followed by Penalized LR and SVM linear at an average accuracy score of 57%. SVM radial has the lowest average score at 51%.

The naive genes dataset achieves an average accuracy range of 48% to 63%, with overall accuracy scores between 35% and 69%. Penalized LR performs relatively well with an average accuracy score of 63%, while random forest has the lowest mean accuracy score at 48%. SVM radial performs at a mid-level with a 59% average accuracy score, and SVM linear slightly outperforms random forest with an average accuracy score of 54%.

For the significant down genes dataset, all classification methods perform performs in a similar fashion, with average accuracy scores ranging from 53% to 57%. Overall accuracy scores are between 41% and 65%. Penalized LR, random forest, and SVM linear achieve average accuracy scores between 55% and 57%, while SVM radial has a slightly lower average accuracy score at 53%.

Finally, the significant up genes exhibit slightly lower performance compared to other gene datasets. Overall accuracy scores range from 27% to 64%, with mean accuracy scores between 43% and 55%. SVM radial shows a slightly better performance with an accuracy score of 55%, followed by random forest and SVM linear with average accuracy scores of 50%. Penalized LR has the lowest average accuracy score at 43%.

In summary, variance filter does not make any significant impact in improving the accuracy scores of classification methods with model up-regulated genes, model down-regulated genes, naive genes, significant up genes and significant up genes datasets. However, the all genes dataset demonstrates promising results, indicating a significant improvement in accuracy.

4.5.2 Permutation filter

A comparative visualization of mean classification accuracy across machine learning models with permutation filter and gene sets, compiled from five experimental iterations is shown in Figure 10 in Appendix on page 53.

The all genes set has the highest accuracy scores and shows the best performance across all other gene sets. The overall accuracy score ranges between 70-83% and the average scores are between 72-79%. Penalized logistic regression and random forest exhibit top performances with around 78-79% mean accuracy scores, trailed by SVM linear and SVM radial having mean accuracy scores of around 76-77%. Overall, the all genes set significantly outperform other sets.

The model down-regulated genes set achieves low overall accuracy scores spread across 27-69%. The mean accuracy scores are around 42-61%. Random forest lags in performance with an average accuracy of around 42%. SVM linear and SVM radial have average accuracy scores of 53% and 50% respectively. Penalized LR has the highest average accuracy of around 60% when compared to other classification methods in model down-regulated genes.

The model up-regulated genes set show a slightly improved average accuracy with 52-63% when compared to model down-regulated genes set. The overall accuracy scores are in the range 42% to 78%. SVM linear exhibits the best performance with an average accuracy score of 63%, followed by SVM linear with 60%. Penalized LR shows mid-level performance with mean accuracy of 57%, while random forest has the least score of around 52%.

The naive genes set displays moderate performance with overall accuracy ranging between 42-81%. The average accuracy scores of all the classification methods are in the range 44-65%. Penalized LR performs noticeably well with a mean accuracy score of around 64%. The average accuracy score of SVM linear is 61%, followed by SVM radial with 58%. Random forest exhibits least performance with an average accuracy score of 49%.

Significant down genes has accuracy scores spread across a wide range of 34-78%. The average accuracy scores are in the range 57-64%. SVM linear has the highest mean accuracy of 64%, while SVM radial has the least accuracy of 57%. The average accuracy scores of penalized LR and random forest are around 61% and 58% respectively.

Significant up genes set shows one of the least performances with the overall accuracy scores ranging between 33-63%, while the mean accuracy scores are in the range 47-58%. Even here random forest performs the least with a mean accuracy of 47%. SVM linear and SVM radial have mean accuracy scores in the range 52-53%. Penalized LR has the highest mean accuracy score of 58%.

Overall, permutation filter on all genes set substantially boosted the performance of the classification methods. The accuracy scores of model up-regulated genes, model down-regulated genes, naive genes, significant up genes and significant up genes do not seem to have improved much even after applying permutation filter.

4.5.3 Impurity Filter

Figure 11 in the Appendix on page 54, provides a comparative visualization of mean classification accuracy across machine learning models on gene sets with impurity filter, compiled from five experimental iterations.

Similar to variance filter and permutation filter, the all genes set achieves the highest performance, with accuracy scores ranging from 75-90% across learners, with average accuracy scores 78-87%. Random forest demonstrates the best accuracy at 87%, while penalized LR also exhibits strong 85% accuracy. Support vector machines have slightly lower mean accuracy around 78-79%. Overall, the all genes set significantly outperform other sets.

For model down-regulated genes, accuracy ranges between 32-74%, with mean accuracy per model across all five experiments is in the range 45-60%. Random forest underperforms with 47% mean accuracy, while SVM radial shows a better performance comparatively with 60% mean accuracy. Penalized LR and SVM linear perform similarly with mean accuracy scores of around 57% and 58% respectively. Performance of model down-regulated genes lags other gene sets considerably.

Model up-regulated genes exhibit similar overall performance as downregulated genes, with overall accuracy scores between 44-68% and mean model accuracy scores across all five experiments is in the range 52-60%. Random forest (Ranger) shows a better performance compared to other models with mean accuracy of around 59%, followed by SVM radial and SVM linear with mean accuracy scores of around 54% and 55% respectively. Penalized LR has the least score of around 52%.

Naive genes enable noticeably better performance when compared to model up-regulated genes, model down-regulated genes, significant up genes and significant down genes. The overall accuracy scores range between 60% to 82% while the mean accuracy is around 70%. Random forest and SVM radial show a good performance with mean accuracy of around 70-71%, while SVM linear and Penalized LR have mean accuracy score of around

68%. Overall, all the models perform quite well with almost similar mean accuracy scores.

Significant down genes exhibit moderate performance with mean accuracy scores ranging from 60-72%. The overall accuracy scores range from 50% to 76%. SVM radial has the highest mean accuracy score of around 72% while penalized LR has the least mean accuracy score of 60%. Random forest also has performed similar to penalized LR with mean accuracy score of around 62%. SVM linear shows a moderate performance with around 65% mean accuracy. Overall, significant down gene set performs better than model down-regulated genes, model up-regulated genes and significant up genes.

Significant up genes perform worst among the sets with overall accuracy scores spread between 36-59%. The mean accuracy scores are around 50-56%. Random forest shows the least performance with a mean accuracy score of 50%. Penalized LR, SVM linear and SVM radial shows similar performance with mean accuracy scores of around 54-56%. Overall, the performance of significant gene set is not so commendable.

In summary, impurity filtering improves accuracy of certain sets, especially significant down and naive genes. But model up-regulated genes, model down-regulated genes and significant up genes still lag behind demonstrating a better performance. The all genes set massively outperforms other gene sets with great accuracy scores.

4.5.4 JMIM filter

Figure 12 in the Appendix on page 55, presents a comparative visualization of mean classification accuracy across various classification methods, utilizing permutation filter and gene sets, derived from five experimental iterations.

The classification methods exhibit slightly enhanced performance when applied to the all genes dataset compared to other datasets. Overall accuracy scores for the all genes dataset range from 53% to 68%, with mean accuracy scores between 56% and 62%. Penalized LR demonstrates a slightly superior performance with an average accuracy score of 62%. Random forest and SVM linear perform comparably, both achieving a 60% average accuracy score, while SVM radial has the lowest score at 56%.

For the model down-regulated genes dataset, the overall accuracy scores vary between 33% and 68%, with mean accuracy scores ranging from 49% to 60%. SVM linear outperforms other methods with an average accuracy score of 60%. Penalized LR and SVM

radial exhibit similar performance, both achieving average accuracy scores of 57% and 58%, respectively. Random forest shows the lowest average accuracy score at 49%.

The performance of classification methods on the model up-regulated genes dataset is slightly lower than that of the model down-regulated genes dataset. Overall accuracy scores for model up-regulated genes range between 36% and 67%, with mean accuracy scores between 45% and 54%. SVM radial attains the highest average accuracy of 54%, followed by SVM linear with a 52% average accuracy. Penalized LR and random forest exhibit similar performance, both achieving an average accuracy of 47.5%.

The naive genes dataset shows varied performance, with overall accuracy scores ranging between 37% and 77%, and average accuracy scores between 43% and 57%. Random forest displays the lowest performance with an average accuracy score of 43%, while SVM linear performs well with the highest score of 57%. Penalized LR achieves a mean accuracy score of 54%, followed by SVM radial with 52%.

The performance of classification methods on significant down genes is similar to that of model up-regulated genes and naive genes datasets. Average accuracy scores range between 43% and 51%, with overall accuracy scores spanning from 32% to 68%. SVM linear and penalized LR exhibit almost identical performance with average accuracy scores of 50-51%. Random forest achieves a 48% average accuracy, while SVM radial shows the lowest performance with an average accuracy score of 43%.

The performance of classification methods onsignificant up genes is slightly lower than that of other datasets. Mean accuracy scores range between 45% and 47%, while overall accuracy scores range from 27% to 60%. Random forest and SVM linear perform similarly, achieving mean accuracy scores of 47-48%, while penalized LR and SVM radial have average accuracy scores of around 45-46%.

In summary, the all genes dataset exhibits slightly better performance than the other datasets. However, the increase in accuracy with JMIM filter is not substantial compared to other filter methods. The JMIM filter does not significantly improve the accuracy of datasets compared to their accuracy with no filter method applied.

5 Summary

This project focused on performing different classification methods to accurately classify tumor development in mice fed with western diet. The dataset comprises gene expression data of mice were fed with both standard diet and high fat diet over a 48-week time course. The downstream objective was to detect tumor vs. non-tumor status, specifically for mice fed a Western diet. A comprehensive approach involving data preprocessing, feature selection, classification methods, and filter methods was undertaken to achieve this goal.

Initially, gene expression data underwent pre-filtering to exclude genes with low read counts, resulting in the removal of 14,994 genes with fewer than 10 total read counts. Subsequently, variance stabilizing transformation was applied to address heteroscedasticity. Once the data was prepared for analysis, differential expression analysis was conducted to assess changes in gene expression over time. Principal component analysis was then performed to visualize the variation between weeks and different diets. Following this, univariate analysis was performed to evaluate the predictive performance of genes by calculating the AUC score for each gene based on the counts of 22 mice and their tumor development labels. A histogram of the AUC scores was generated, and a permutation analysis was conducted to compare the results to random expectations. The analysis concluded that the genes lacked strong predictive power for the given tumor development labels.

Later, five feature selection methods were employed to identify subsets of the most informative genes from the Standard diet. This led to the identification of significant up-regulated genes (844), significant down-regulated genes (871), naive genes (500), model down-regulated genes (500), and model up-regulated genes (362).

Following feature selection, classification methods such as Random Forests, SVM with a linear kernel, SVM with a radial kernel, and Penalized Logistic Regression were implemented to classify mice with tumor and those without. For all the classification methods, nested cross-validation was utilized, and hyperparameters were optimized through auto-tuning.

The accuracy scores and confusion matrices from all the classification methods were thoroughly discussed and analyzed. Random forest performed poorly by achieving only around 47% to 52% mean accuracy across all the datasets. SVM linear achieved a slightly improved accuracy in the range of 50% to 62%. SVM radial exhibited moderate

performance with average accuracy scores in the range 49% to 57%. Penalized LR achieved average accuracy scores between 46% to 57%.

The overall average accuracy ranged substantially from 46-62%, however the mean was only around 52%. This suggested that tumor development prediction from gene data was generally mediocre across the methodologies explored. No individual model significantly and consistently outperformed others. Random Forest, Support Vector Machines, and Penalized Logistic Regression all exhibited similar promises and pitfalls, with mean accuracy scores varying little between classification algorithms.

Lastly, filter methods were implemented to assess whether they could enhance the accuracy scores of classification methods. Various filter methods, including variance, permutation, impurity, and JMIM filters, were applied to all genes without prior feature selection, as well as to the datasets selected from feature selection.

Filter methods significantly enhanced the accuracy of the all genes dataset. The Impurity filter outperformed all other filter methods when applied to the all genes dataset, by achieving average accuracy scores within the impressive range of 78-89% across all classification methods. Following closely was the permutation filter, exhibiting a mean accuracy score of 72-79%. Similarly, the Variance filter demonstrated competitive performance, with average accuracy scores spanning from 68% to 85%. In contrast, the JMIM filter displayed the lowest performance, yielding mean accuracy scores ranging from 56-62%. In terms of overall performance across the Variance, Permutation, and Impurity filter methods, Random Forest stood out as the most effective. It attained the highest mean accuracy score of 87% with the Impurity filter. Notably, the filter methods did not introduce any significant differences to the accuracy scores of the already selected feature sets.

In summary, the classification methods achieved only approximately 50% average accuracy, failing to reliably categorize mouse tumor development status. Potential contributing factors might include focusing solely on standard diet data for initial feature selection as well as the limited sample size of 22 labeled mice. However, notable progress emerged when filter methods were applied to all genes, resulting in a substantial improvement in accuracy across various classification methods. To enhance the project further, future endeavors could focus on refining feature selection strategies and fine-tuning classification models even more, paving the way for more accurate predictions in tumor development.

Bibliography

Becker, M., Lang, M., Richter, J., Bischl, B. and Schalk, D. (2023), 'mlr3tuning: Hyperparameter Optimization for 'mlr3'', `https://mlr3tuning.mlr-org.com/`. [Online; accessed November-2023].

Binder, M., Pfisterer, F., Schneider, L., Bischl, B., Lang, M., Fischer, S. and Dandl, S. (2023), 'Preprocessing operators and pipelines for 'mlr3".

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A.-L., Deng, D. and Lindauer, M. (2021), *Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges*.

Bischl, B., Sonabend, R., Kotthoff, L. and Lang, M. (2023), 'Applied Machine Learning Using mlr3 in R', `https://mlr3book.mlr-org.com/`. [Online; accessed November-2023].

Bommert, A., Sun, X., Bischl, B., Rahnenführer, J. and Lang, M. (2020), *Benchmark for filter methods for feature selection in high-dimensional classification data*.

Ghallab, A., Myllys, M., Friebel, A., Duda, J., Edlund, K., Halilbasic, E., Vucur, M., Hobloss, Z., Brackhagen, L., Begher-Tibbe, B., Hassan, R., Burke, M., Genc, E., Frohwein, L. J., Hofmann, U., Holland, C. H., González, D., Keller, M., latif Seddek, A., Abbas, T., Mohammed, E. S. I., Teufel, A., Itzel, T., Metzler, S., Marchan, R., Cadenas, C., Watzl, C., Nitsche, M. A., Kappenberg, F., Luedde, T., Longerich, T., Rahnenfuhrer, J., Hoehme, S., Trauner, M. and Hengstler, J. G. (2021), *Spatio-Temporal Multiscale Analysis of Western Diet-Fed Mice Reveals a Translationally Relevant Sequence of Events during NAFLD Progression*.

Han, J., Kamber, M. and Pei, J. (2012), *Data Mining Concepts and Techniques*, third edn, [Morgan Kaufmann Publishers], 225 Wyman Street, Waltham, MA 02451, USA.

Hastie, T., Tibshirani, R. and Friedman, J. (2008), *The Elements of Statistical Learning*, New York.

Henry, L., Wickham, H., François, R., Müller, K. and Vaughan, D. (2023), 'A Grammar of Data Manipulation', `https://cran.r-project.org/web/packages/drc/index.html`. [Online; accessed November-2023].

Hoerl, A. E. and Kennard., R. W. (1970), *Ridge Regression: Biased Estimation for Nonorthogonal Problems*.

James, G., Witten, D., Hastie, T., Tibshirani, R. and Taylor, J. (2009), *An Introduction to Statistical Learning*, New York.

Jolliffe, I. T. (1986), *Principal Component Analysis*, [Springer series in statistics], New York.

Kotu, V. and Deshpande, B. (2015), *Predictive Analytics and Data Mining*.

Kursa, M. B. (2022), 'Tools for information-based feature selection and scoring', `https://cran.r-project.org/web/packages/e1071/index.html`. [Online; accessed November-2023].

Lang, M., Au, Q., Coors, S. and Schratz, P. (2023), 'Recommended Learners for 'mlr3'', `https://mlr3learners.mlr-org.com/`. [Online; accessed November-2023].

Liang, Y., Liu, C., andKwong Sak Leung, X.-Z. L., Chan, T.-M., Xu, Z.-B. and Zhang, H. (2013), *Sparse logistic regression with a L-1/2 penalty for gene selection in cancer classification*.

Love, M., Huber, W. and Anders, S. (2014), *Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2*.

Love, M. I., Huber, W. and Anders, S. (2023), 'Analyzing RNA-seq data with DESeq2', `https://bioconductor.org/packages/devel/bioc/vignettes/DESeq2/inst/doc/DESeq2.html`. [Online; accessed October-2023].

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., Chang, C.-C. and Lin, C.-C. (2023), ' Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien', `https://cran.r-project.org/web/packages/e1071/index.html`. [Online; accessed November-2023].

R Development Core Team (2013), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

Ritz, C., Jensen, S. M., Gerhard, D. and Streibig, J. C. (2015), *Dose-Response Analysis Using R*.

Ritz, C. and Strebig, J. C. (2016), 'Analysis of Dose-Response Curves', https://cran.r-project.org/web/packages/drc/index.html. [Online; accessed November-2023].

Schratz, P., Lang, M., Bischl, B. and Binder, M. (2023), 'Filter Based Feature Selection for 'mlr3'', https://mlr3filters.mlr-org.com/. [Online; accessed November-2023].

Tibshirani, R. (1996), *Regression Shrinkage and Selection Via the Lasso*.

Wainer, J. and Cawley, G. (2021), *Nested cross-validation when selecting classifiers is overzealous for most practical applications*.

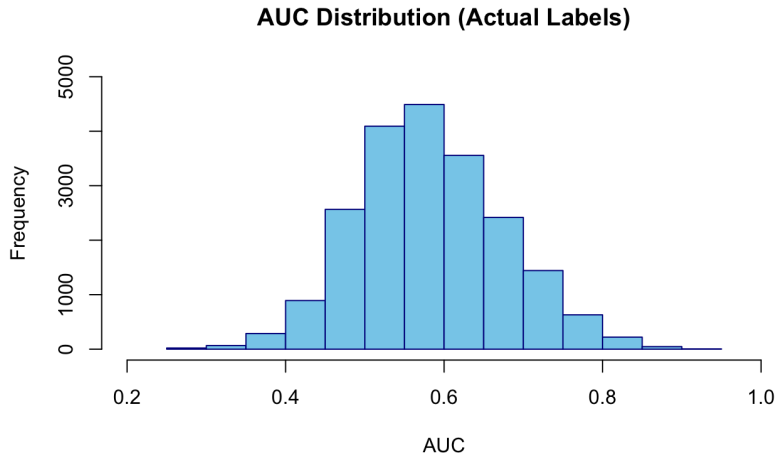
Wickham, H., Chang, W., Henry, L., Pedersen, T. L., Takahashi, K., Wilke, C., Woo, K., Yutani, H. and Dunnington, D. (2023), 'Create Elegant Data Visualisations Using the Grammar of Graphics', https://ggplot2.tidyverse.org/. [Online; accessed November-2023].

Wright, M. N., Wager, S. and Probst, P. (2023), ' A Fast Implementation of Random Forests', https://cran.r-project.org/web/packages/ranger/index.html. [Online; accessed November-2023].

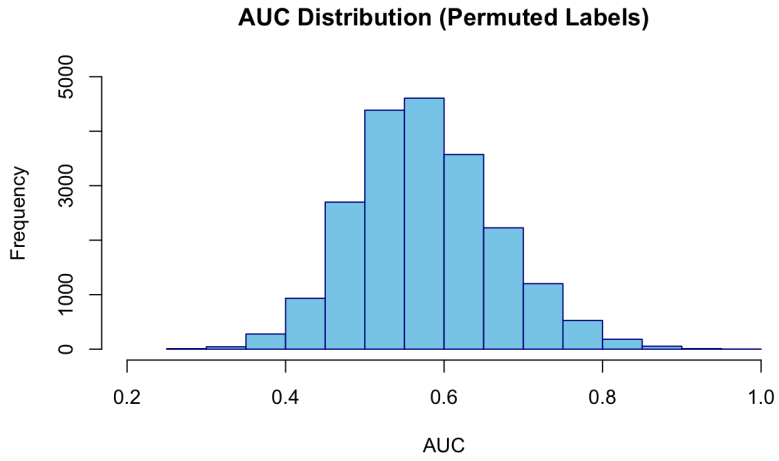
Zou, H. and Hastie, T. (2005), *Regularization and variable selection via theelastic net*.

Appendix

A Additional figures



(a) Actual labels



(b) Permuted labels

Figure 8: AUC distribution with actual and permuted labels

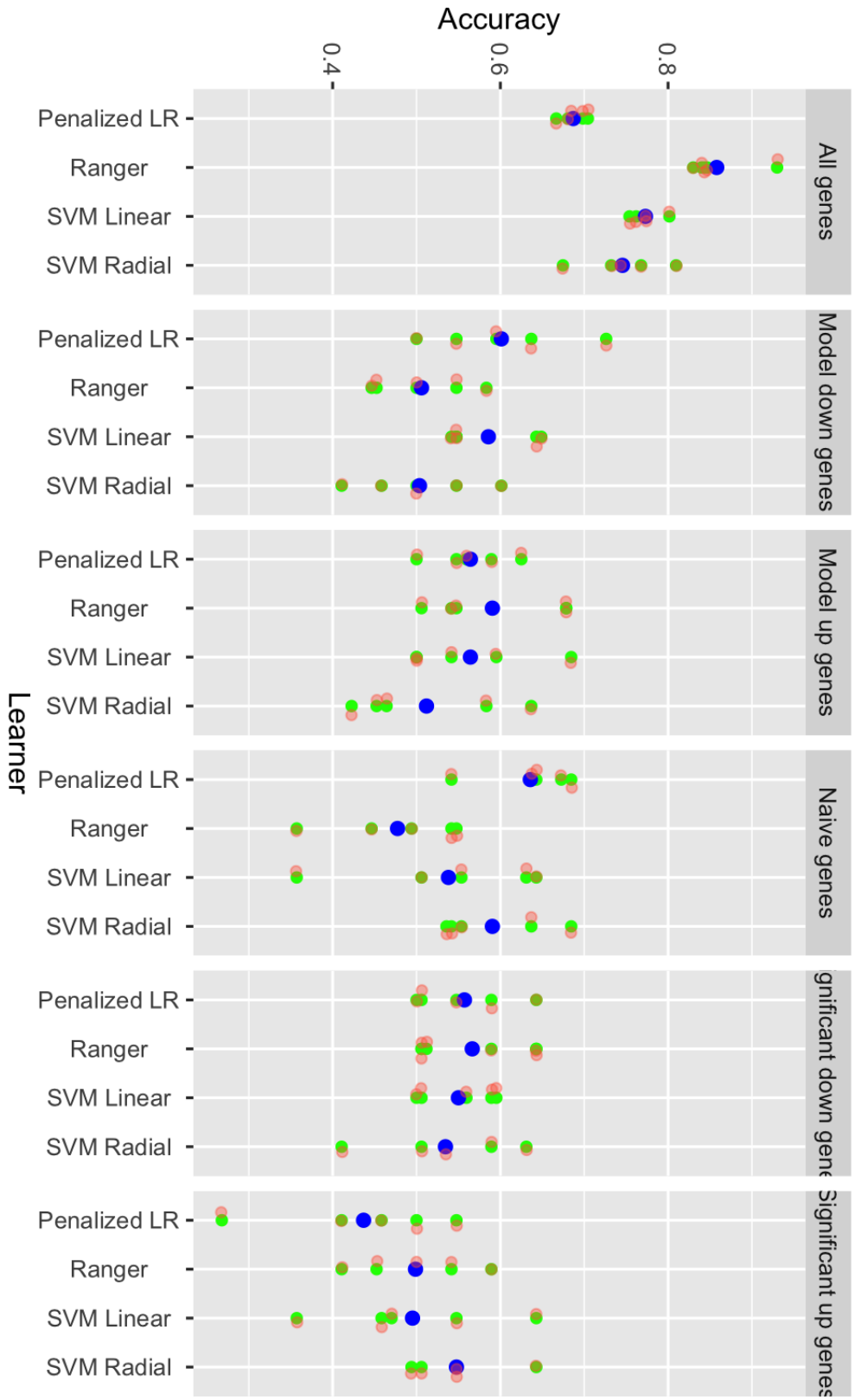


Figure 9: Accuracy by Learner and Dataset with Variance filter

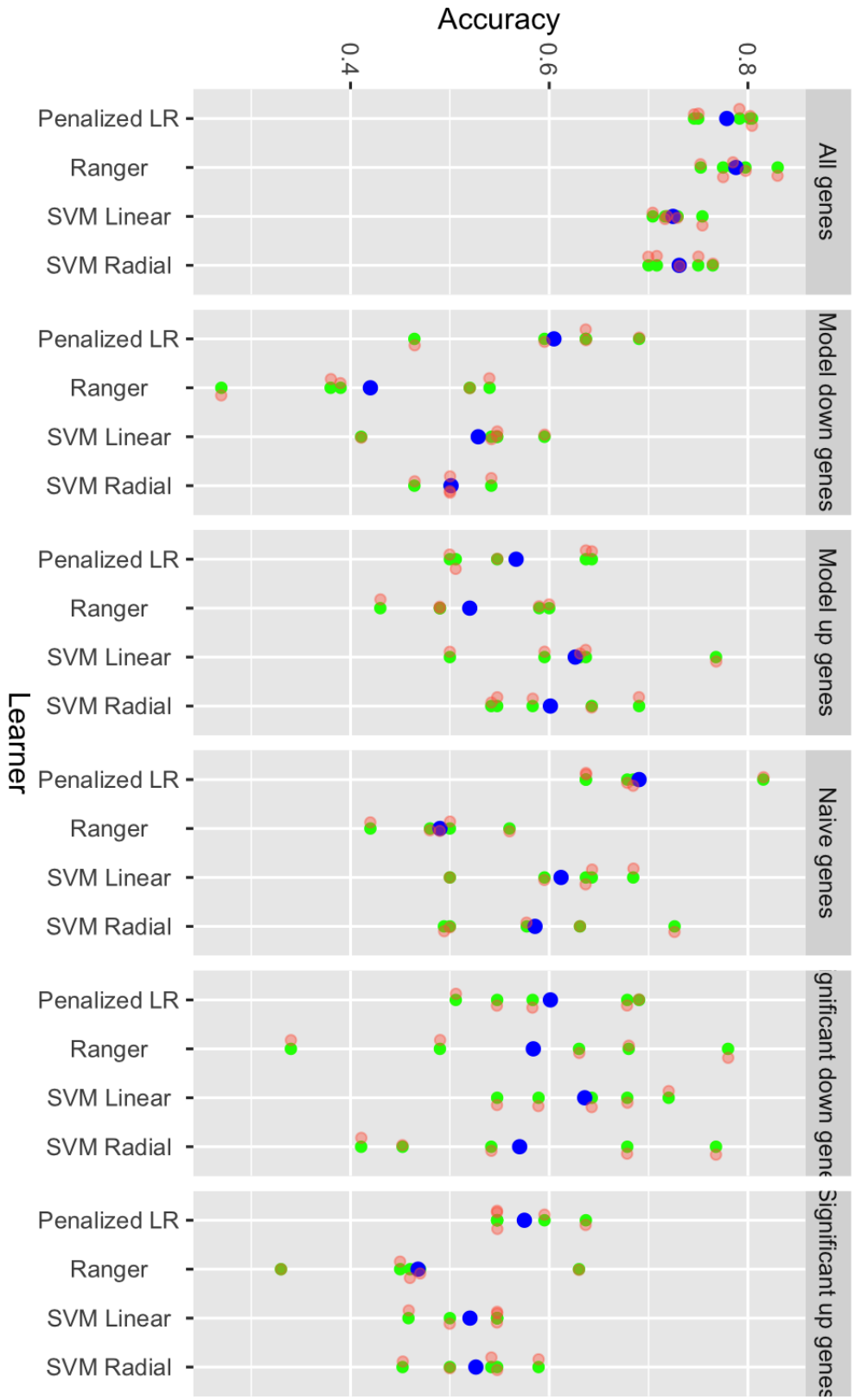


Figure 10: Accuracy by Learner and Dataset with Permutation filter

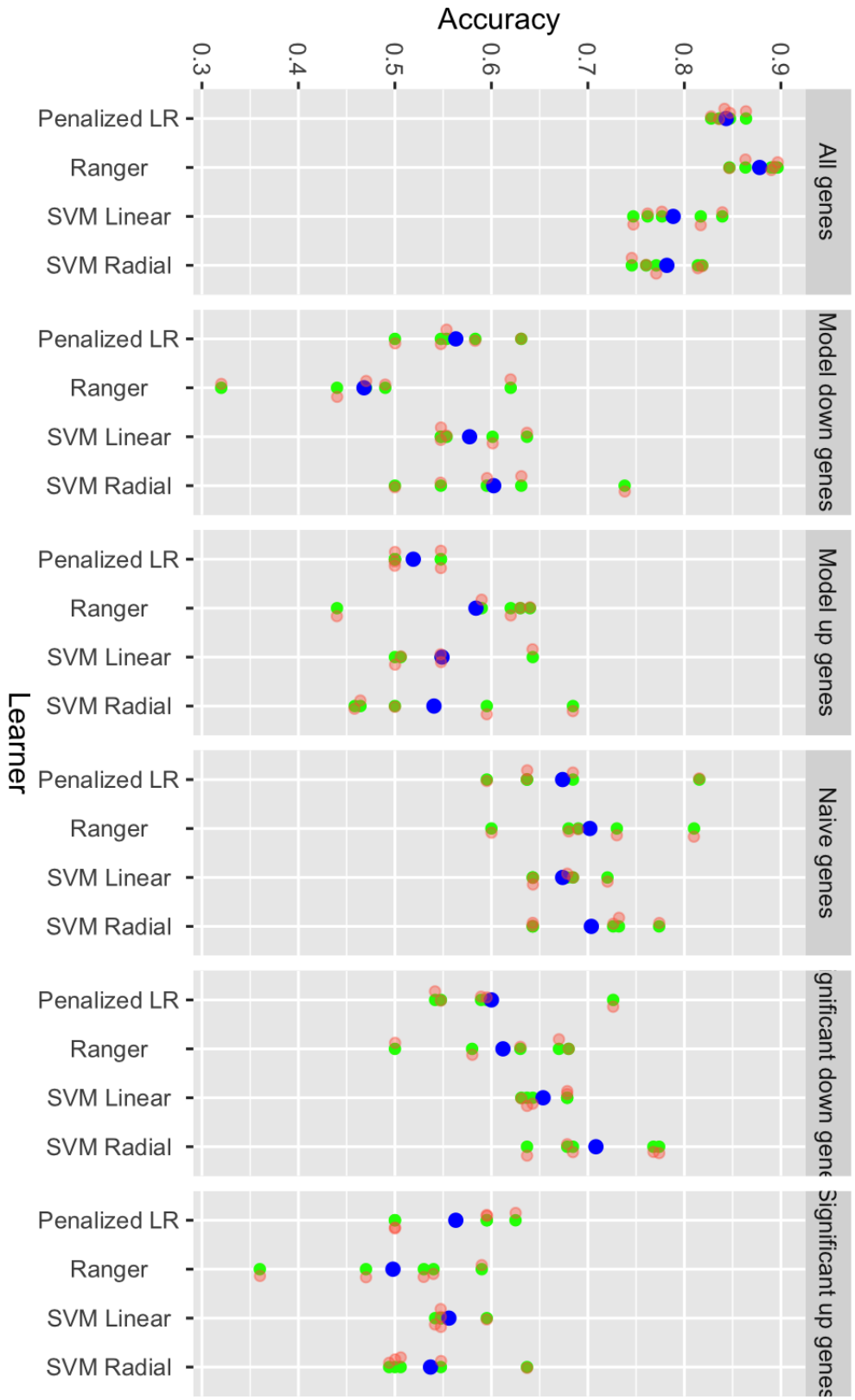


Figure 11: Accuracy by Learner and Dataset with Impurity filter

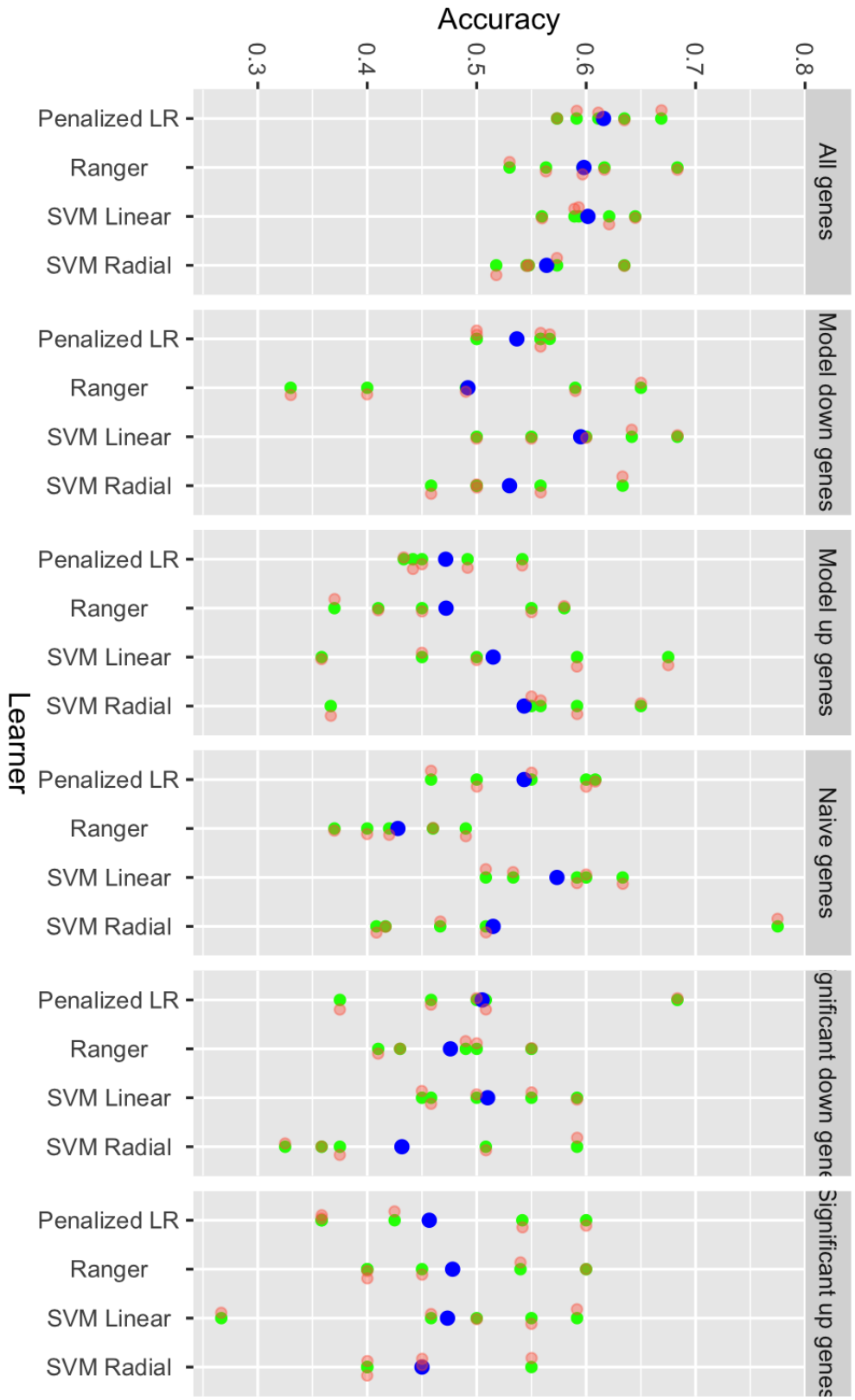


Figure 12: Accuracy by Learner and Dataset with JMIM filter