IDENTIFYING BIOMARKERS FOR IRINOTECAN SENSITIVITY & PREDICTING RESPONSIVE PATIENT CLUSTERS

Bilge Elitok
M.Sc. Student of Computational Biomedicine
University of Eastern Finland
belitok@uef.fi

December 30, 2024

ABSTRACT

The project investigates how gene expression data from cancer cell lines can predict patient responses to irinotecan, a chemotherapy drug. Using colon cancer cell line data, biomarkers related to irinotecan sensitivity were identified through regularization-based methods such as Ridge and ElasticNet regression. These biomarkers were then used to cluster colon cancer patient data, aiming to identify subgroups with distinct treatment responses. The results revealed that higher WRAP53 (WD40 Repeat-Containing Protein Antisense To TP53) expression correlates with irinotecan sensitivity in certain patient clusters ($p = 1.73 \times 10^{-6}$). Another candidate biomarker MAP3K4(Mitogen-activated protein kinase kinase kinase 4), once significant for cell line subgroups show no distinctive capability between colon adenocarcinoma patients, highlighting the gap between cancer models and patient data, as well as the importance of validating biomarkers discovered in cell lines on patient datasets for clinical applicability.

1 Introduction

One of the goals of precision medicine is to refine treatment strategies tailored by the pharmacogenomic response of the individual. Cancer cell lines, besides being the most widely used models for cancer biology and drug response, provide a controlled environment to study gene expression patterns influencing availability of a drug for the individual's system. Two key resources for cataloging pharmacogenomic profiles in cancer cell lines are the Cancer Cell Line Encyclopedia (CCLE) and The Genomics of Drug Sensitivity in Cancer (GDSC) (Nusinow et al., 2020; Yang et al., 2013). These projects bridges the alterations in gene expression to drug response, thereby facilitates discovery of cancer biomarkers that influence treatment decisions, highlighting the underlying the transcriptomic heterogeneity of human cancers that influences patient responses.

This project aims to investigate how gene expression matrices extracted from cancer cell lines can be leveraged to predict drug response clusters among data from colon cancer patients, and to identify the cluster of patients that respond well to cancer drug, irinotecan.

For this aim, the project was implemented in two steps:

1. **Biomarker Selection:** Using regularization-based methods like Ridge and ElasticNet alongside Linear Regression, it was aimed to identify the most predictive biomarkers for irinotecan sensitivity through comparison across methods.
2. **Patient Clustering:** Using the Colon Adenocarcinoma (COAD) dataset, we cluster patients based on gene expression profiles and test biomarker expression levels for prognostic significance.

The results and predictive power of the biomarkers are evaluated by comparing their relevance in cancer cell line models and colon adenocarcinoma patient samples.

2 Datasets and Environment

CCLE dataset is used for irinotecan screening was used for selecting the model, training and hyperparameter tuning. Then, GDSC dataset served as an external dataset for final model evaluation. The target columns in both CCLE and GDSC datasets are the AUC (Area Under the Curve) values, which represent the response to irinotecan with higher AUC values indicating greater sensitivity. Both datasets share overlapping pharmacogenomic features but are implemented independently, i.e. no part of GDSC was seen by the model during training and tuning.

For the second part, the COAD dataset is used for patient clustering and biomarker evaluation based on gene expression profiles. COAD dataset is gene expression data from Colon Adenocarcinoma (COAD) patients obtained from the TCGA-COAD database.

2.1 Dependencies

Before preparation of the the data, libraries central to preprocessing, exploratory data analysis and machine learning were imported. Dependency list with example functions are the following:

- **pandas (pd)**: Data manipulation and analysis. (Example functions: `DataFrame`)
- **numpy (np)**: Numerical computations and array handling (Example functions: `percentile`).
- **random** Python's own random was imported for setting random seeds for reproducibility.
- **requests**: Used for fetching gene names by ensembl IDs, through Ensembly API.
- **OS (os)**: File and path management (Example functions: `path`)
- **Matplotlib (plt)**: Visualization (Example functions: `scatter`)
- **Seaborn (sns)**: Data visualization and custom plotting (Example functions: `heatmap`)
- **SciPy (scipy)**: Statistical analysis (Example functions: `shapiro, MannWhitneyU`)
- **requests**: HTTP requests for fetching external data from Ensembl API. (Example functions: `get, json`)
- **Scikit-Learn)**: Machine learning models, data pre-processing, and evaluation (Example functions: `PowerTransformer, KMeans, SimpleImputer,GridsearchCV`)
- **%matplotlib inline**: Ensures plots displayed within the Jupyter Notebook

2.2 Inpecting the Contents of the Datasets

As the first step of data analysis, the basic properties of datasets e.g. dimensions, whether an ID column exists, the datatype for target column was inspected. Both CCLE and GDSC datasets contains names of the cell line models as string values along the first column named "Unnamed :0". Feature columns includes gene expression information, essentially the mRNA count for the cell line. Feature columns are titled by the ensemble gene codes for each gene, therefore a general pattern in the form of "ENSG " that identifies a feature column can be found. Target column for both CCLE and GDSC datasets contains contains, the continuous target variable AUC (Area Under Curve), which reresents the sensitivity for the each the cell line to the chemotherapy drug irinotecan. AUC quantifies the efficiency of irinotecan on a given cell line. Higher AUC values indicate sensitivity, a more effective response to irinotecan while the lower AUC values signify resistance, suggesting that the cell line exhibits a weaker response to irinotecan.

The raw data was stored in local machine as a csv file, thus, a custom function, `load_data()`, to load data using their path information and returns the summary of key information was defined.

The function globalizes the .csv files and stores them in object defined by their base names (i.e. name without the folder suffix). Running `load_data()` function returned three pandas dataframe objests and a summary dataframe:

	File Name	Row Count	Column Count	Duplicates	Has Null	Total Null	30% missing?
0	ccle_irinotecan_mrna	309	496	0	True	1515	[]
1	test_irinotecan_mrna	535	496	0	False	0	[]
2	ccle_irinotecan_muta	293	64	0	False	0	[]

Table 1: Summary of dataset characteristics.

Contents of the datasets are inspected for their dimensions, missing values and duplicate values. CCLE and CDSC datasets are appeared to contain same number of features but different number of cell lines.

Upon inspection, only 'ccle_irinotecan_mrna' dataset appeared to have features with null values, however none of those features has more than 12% among the column missing. For the better insight of the distribution charachteristics of features and the target, a custom function `plot_skewness_and_target_distribution()` was used. This function takes dataset and the name of the target column as the input. Then the function creates barplots for the skew values for each of the feature along with the histogram of the target distribution for the dataset. The histogram of the target column includes Kernel Density Estimation (KDE) curve as a continuous line. KDE curve represents the continuous estimate of the target variable's distribution, helps visualisation of data points' concentration across different target values.

The first column, involves the names for the cell lines as string were dropped, as numerical data is needed for the downstream steps.

For all of the datasets, the target column is a continuous AUC value. For downstream steps involving classification and analysis of different classifiers, a function to categorize this numerical target variable was defined: `sort_sensitivity_class()`.

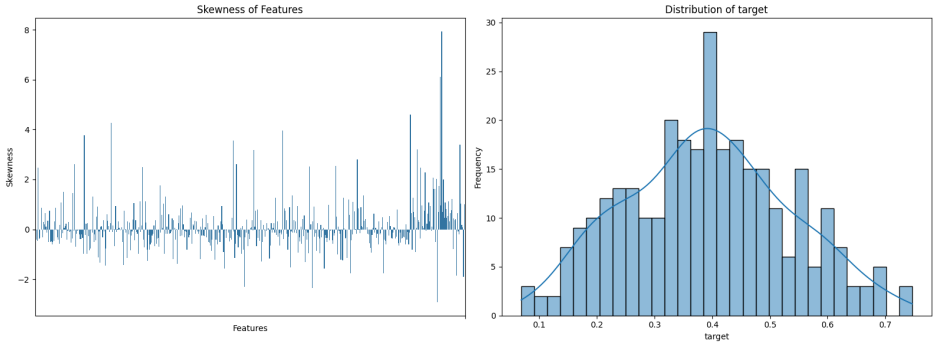

Figure 1: Skewness of Features and Distribution of the Target variable for CCLE dataset, i.e. mRNA expression matrix for different cell lines models and sensitivity to irinotecan as target variable.

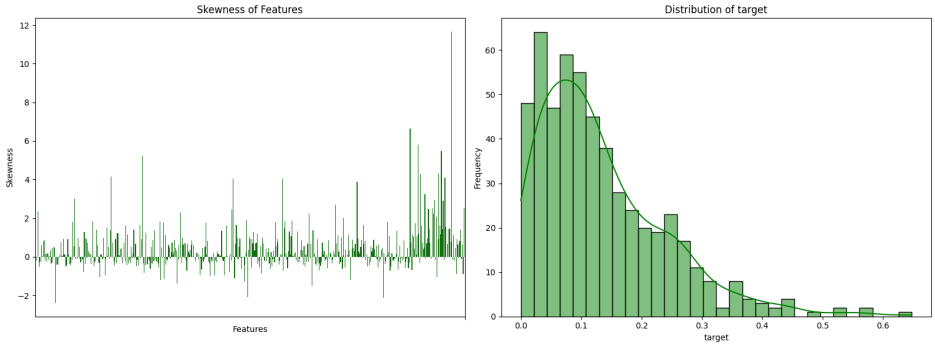

Figure 2: Similar to figure 1, the skewness of Features and Distribution of the Target variable for CCLE dataset, i.e. mRNA expression matrix for different cell lines models and sensitivity to irinotecan as target variable is visualized.

For both figure 1 and 2, features with high positive skewness suggest that their divergence from normal distribution proportional to their skew value. For higher skew values, distributions are more concentrated towards the lower end and have a long tail towards the higher values. Skewed distributions are commonly encountered in real-world datasets, especially when dealing with biological or clinical data, where most samples are concentrated in one of the ends, but a few extreme values may occur on the opposite end due to outliers or rare events.

This function accepts a data object contains a 'target' column which holds the continuous values to be classified. In order to specify the threshold for different categories, the target values are divided into three parts:

- Bottom 33 percentile, low AUC values indicating **resistance,** numerically encoded as 0,
- Top 33 percentile, high AUC values indicating **sensitivity,** numerically encoded as 2,
- Every value inbetween these two ends were accepted as having **mild response** and encoded with 1.

When comparing the data sets, the CCLE dataset appeared to have a middle range located at 0.335–0.450, and test dataset to have a middle range located at 0.074 - 0.149. This suggests, overall, ccle has a higher range of expression values than that of test.

For CCLE dataset the difference between the lower and higher percentiles (0.34 and 0.45, respectively) is relatively small (0.115). This finding suggests the central 33% of the data points are relatively tightly clustered in terms of value. Test dataset shows even more narrowly distributed values around the central.

Feature Scaling: Normalization or Standardization?

This section is based on Scikit-Learn Documentation 6.3 Preprocessing Data

In order to ensure all features are able to contribute to the learning process of the model in the equal scale, the values of features needs to be aligned. This is achieved by transformation of the values in a similar scale for compatibility of the range across features(Hastie et al., 2009) . This practice falls into the category of feature scaling, commonly achieved by normalization or standardization of the features.

If the features are **measured in different scales**, for example, house price in dollars and house size in meter squares for a house price prediction task, they need to be adjusted to a common scale. Main types of normalization include: **Log Normalization** which reduces the impact of larger values, **Mean Normalization** which shows how each value compares to the average and **Min-Max Normalization** which rescales the values to fit a range of 0-1.

Also called Z-Score Transformation, Standardization transforms the data to have zero-mean ($\mu = 0$) and unit-standard deviation ($\sigma = 1$) distribution(Aho, 2014).

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{1}$$

Both CCLE and GDSC datasets involves gene expression profiling combined with irinotecan dose responses per cancer line. Unlike the house prices example above, gene expression dataset are not measured in different scales, instead the expression intensity might be measured within a microarray, or in the case of single cell RNA sequencing, essentially mRNA molecules are counted. This aspect of the dataset pointed out Z-score transformation as a preferable method. Upon Z-score standardization, feature values shows relative change in expression for each gene.

Another important aspect to consider is **the difference in the overall expression levels between CCLE and GDSC datasets** which is shown in figure 3. Unprocessed GDSC have overall much lower mean expression than unprocessed CCLE. The difference between overall expression levels might be due to batch effects, where variation is the result of technical sources conducting gene expression analyses (Espín-Pérez et al., 2018). Both datasets are large, their preparation took long periods of time and data preparation was performed in lots, therefore has differential measurement errors or non-biological variations across different batches.

Visualizing gene expression profile of cancer cell lines (CCLE and GDSC)

For visualising gene expression data with target variable, a custom function, `heatmap_sensitivity`, based on Seaborn's `heatmap()` was defined. For this purpose, heatmap with a `coolwarm` color scheme was selected to convey differences in gene expression levels efficiently, with higher gene expression was represented by the warmer colors. Inversely, the colors gets colder for lower levels of gene expression.

For visualizing target variable along with gene expression matrix, **sensitivity bars** were added. Sensitivity bars represents the category of the target variable where red indicates resistance, green sensitivity and yellow mild response. For the corresponding categories of the target variable, the sensitivity discretization function, `sort_sensitivity_class()`, which was defined earlier, was used.

For the large scale gene expression matrices, seaborn's heatmap was made bases for visualization. Including the target variable, the drug sensitivity to heatmas was a challenge to overcome at this step. By making use of sensitivity classes, each of the cell_lines were featurazied by a overlayed colorbar. Green being more sensistive, yellow as mild and the red as resistant.

When compared side by side, the overall expression of level among GDSC was appeared to be lower than CCLE, as colder colors was majority through the heatmap.

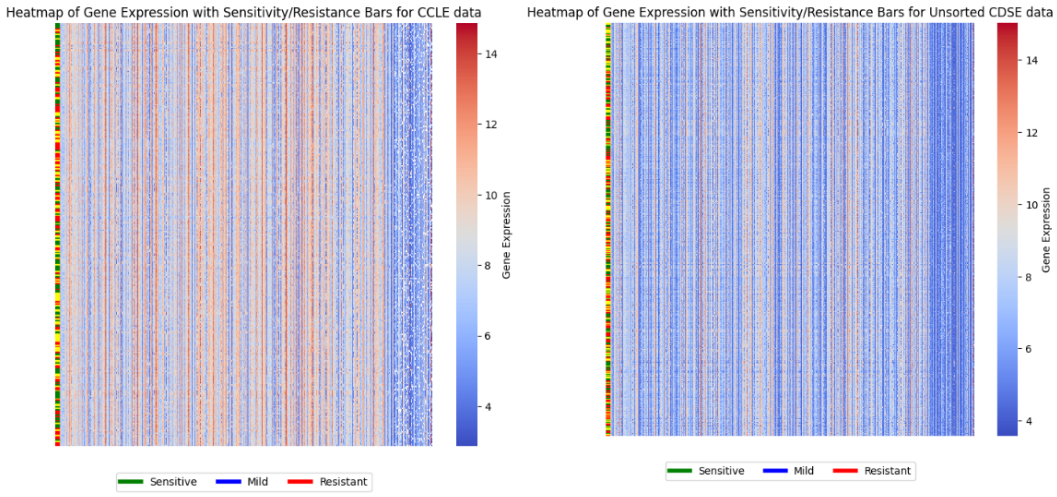

Figure 3: Heatmaps for Gene Expression Matrix for the Cell Lines (rows) and the Genes (columns) included in CCLE (left) and GDSC (right) datasets. Warmer colors indicate higher expression while the colder colors indicate lower expression. The target variable, assigned sensitivity category was represented as bars overlaid at the start of the columns.

Another exploration was made possible at this step. The distribution of the three categories. This was allowed by adding only one line to the sorting algorithm. And as the result, equal distribution for all three of the classes were also visualized with the heatmap. This finding as heatmap representations of gene expression were appeared to be in line with the value distribution among percentile ranges calculated for both datasets.

Next, the null values in CCLE dataset (as it was reported by the summary charachteristics, table 1) were appeared to be concentrated on the final columns. Final columns of CCLE dataset on figure 3 showed sparsity among final columns where zero expressions appeared as white spaces.

2.3 Discretization of Irinotecan Sensitivity into Three Response Groups

Sensitivity categories are also counted by defining a count function, `count_sensitivity_categories()` and returning a counts summary. This function includes a dictionary that defines the labels corresponding to different sensitivity levels. The keys (0, 1, 2) represent numerical encoding of the categories compatible with the that of the the previously defined function `sort_sensitivity_class()`, and the corresponding values ('Resistant', 'Mild', 'Sensitive', respectively) describe the numerical encoding.

Category	CCLE	CDSE
Resistant	105	182
Mild	102	177
Sensitive	102	176

Table 2: Snapshot of the comparison table that shows the count of the instances in three categories with numerical encodings: 0 for Resistance, 1 for Mild Response, 2 for Sensitivity to Irinotecan.

2.4 Normality Testing and Transformation

Inspection of the datasets were followed by statistical testing for normality as certain ML algorithms (e.g. linear regression) and imputation methods (e.g. mean imputation) assumes data is drawn from normal distribution. In the normality test, Shapiro-Wilk test, the null hypothesis is that the data is normally distributed, and the alternative hypothesis is that the data is not. The Shapiro-Wilk statistic is calculated by comparing the ordered values of the sample data with the expected values from a normal distribution as follows (Shapiro & Wilk, 1965):

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_i\right)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{2}$$

Where:

- W is the Shapiro-Wilk test statistic, used to assess normality of the data.
- x_i represents the individual samples
- $\bar{x}$ is the sample mean of the data
- a_i are the coefficients derived from the normal distribution

Whether the assumption of normality is satisfied for the features in CCLE dataset was determined.. A high W value (close to 1) suggests that the data follows a normal distribution while A low W value (significantly less than 1) suggests that the data deviates from normal distribution. Similarly , for p < 0.05 null hypothesis rejected, indicating the data is not normally distributed.

Dataset	Gaussian Columns	Non-Gaussian Columns
CCLE	207	287
Test	99	395

Table 3: Distribution of the count of columns (features) that follow either Gaussian or Non-Gaussian distributions in CCLE and Test Datasets

To evaluate the distributional characteristics of each feature, the Shapiro-Wilk test was applied to each column. The Shapiro-Wilk test evaluates whether the data is drawn from normal distribution by generating a W-statistic and a corresponding p-value. As mentioned higher values for both W-statistic and p indicates closer alignment with Gaussian distribution. For that the custom function, show_distributions() that categorized the features based on their p-values reports the number of Gaussian and non-Gaussian.

For CCLE dataset, several features indicated significant deviation from normality. Almost 68% of features shown deviation from normality. These results indicated the need of transformation techniques further.

3 PCA and Multicolinearity Assessment

This section is based on chapter 6. Linear Model Selection and Regularization from An Introduction to Statistical Learning by James et al., (2013)

Principle Component Analysis (PCA), is a statistical procedure based on orthogonal transformation that converts correlated variables into a set of uncorrelated principle components. First principle component is defined to account for as much of the variability as possible. PCA is sensitive to feature scaling, as features with large scale tend to have larger variance. This renders scaling the data is necessary therefore data is standardized to have a zero-mean and unit variance via `StandartScaler()`.

Both of the CCLE and GDSC datasets are high dimensional with $N = 494$ covariates, meaning that data points are spreaded out. Sparsity distilles out the impact of truly predictive variables and increases the risk of overfitting, where the model learns noise and random fluctuations rather than generalizable patterns.

PCA was applied for in order to interpret correlations between variables and the visualize the explained variance by the principle components.

3.1 Explained Variance Ratio and First Two Principle Components

Scikit-learn's `PCA()` class takes either number of components of the desired explained variance for `n_components=` argument. PCA with 10 principle components were applied to both datasets.

First, PCA was used for dimensionality reduction, reporting a specified number of main components that explain the variance in the data. Then the first two principal components served as the basis for the data separation. Class separability on the bases of first two principle components were assessed by two plots: an explained variance ratio for each principal component and a 2D scatter plot of the first two principal components.

For this purpose a custom function `plot_pca` was designed. This function splits datasets, performs PCA and visualizes results. It accepts dataset and number of principle components (or the amount of explained variance) as inputs. Function also has a mode argument to control target discretization. Function works for both discrete and continous values of y, and guides through the scatter plot with a legend or a colorbar.

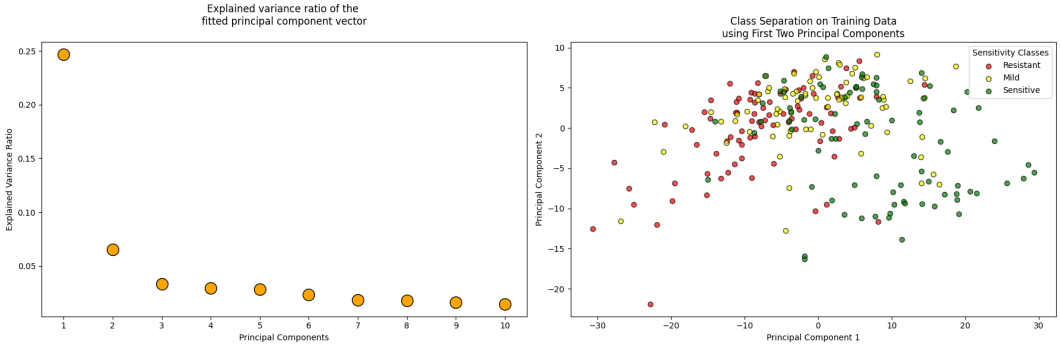

Figure 4: Explained variance ratio of the ten principal components and class separation based on the first two of them for ccle_irinotecan_mrna dataset.

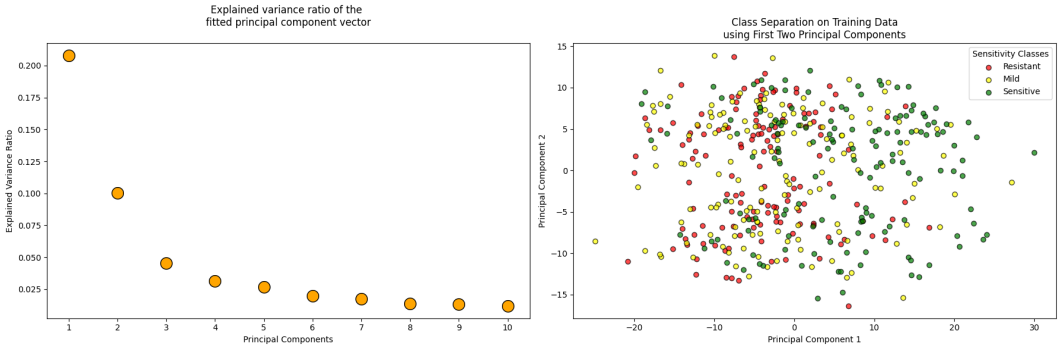

Figure 5: Explained variance ratio of the ten principal components and class separation based on the first two of them for test_irinotecan_mrna dataset.

3.2 Colinearity Assessment among Features of Datasets

For our gene expression data with 494 features, variance tends to be distributed across many dimensions rather than being concentrated around a few features. For CCLE dataset, the first principle component explains almost 25% or the variance across dataset, and for GDSC it explains slightly more than 20%. This result indicates that, for our gene expression datasets, there are no single direction or even small number of directions available to capture the majority of the variance in the datasets.

As only 20 to 25 % variance is explained by first two principle componeents, reducing dimensionality with PCA would result in significant information loss. This result is supported by the class separation plots for both datasets, where sensitivity clusters appeared to be non-separable based on first two principle components (*fig. 4 and fig.5, scatter plots on the right hand side*).

Multicolinearity

As gene expression matrices are often high-dimensional datasets, where number of features are equal or higher than the number of observations. In such cases, the dataset may suffer from multicolinearity, a condition where some of the features are highly correlated with each other. This might lead certain models to struggle as their assumption of feature independence is violated in this case, leading to poor quality estimates. Furthermore, the signal from truly predictive biomarkers are diluted by those redundant features. And without an efficient feature selection strategy, downstream analysis are placed in risk of by models with biases, overfitting and poor generalization abilities.

Therefore, to ensure the retainment of relevant features, multicolinearity assesment is essential. Regularization based methods, such as Ridge, Lasso and Elastic Net is especially valuable for mitigating collinearity by introducing penalties that constraint coefficient estimates (James et al., 2013).

First, correlation matrices for features were visualized by setting the thresholds 0.25 and 1.00. The lack of strong correlations are represented with heatmaps of figure 6.

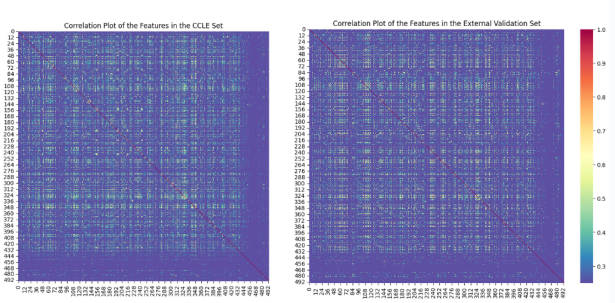

Figure 6: Correlation heatmaps showing the pairwise relationships between features in the CCLE and GDSC datasets. The color intensity represents the strength of the correlation, with warmer colors indicating stronger positive correlations and cooler colors indicating negative correlations. Most correlations are weak to moderate, with the majority falling within the range of [-0.5, 0.5].

For CCLE dataset, the average correlation coefficient was r = 0.09 ± 0.24, with 97% of the coefficients within the range [-0.5, 0.5]. For GDSC, The average correlation coefficient was r = 0.07 ± 0.23, with 96% of the coefficients within the range [-0.5, 0.5].

A correlation coefficient of 0.5 is moderate and correlation between variables only said to be strong if for correlation coefficients in the rande of 0.85 to 1.00. Therefore, most of the pairs of features of both datasets considered to have weak to moderate correlations.

4 Splitting the Datasets, Imputation, Scaling and Transformation

While preparing data for ML models, several preprocessing steps are required: dataset splitting, handling missing values, scaling and transformation of features. These essential steps ensure data is in a compatible format and follows a distribution that is valid for the model's assumptions.

Appropriate handling of these preprocessing steps are crucial to avoid introducing biases to model or overfitting. One particular issue to avoid during preprocessing is data leakage, where information from the test set leaks to the fitting process of the model, and influences training. Data leakage also impairs the integrity of the performance evaluation, leading to overly optimisitic evaluation of the model and to misleading conclusions about the model's true performance.

In order to address this challenge, a key strategy based on fold-wise processing and cross-validation was applied. This strategy aimed to ensure that each preprocessing task, such as imputation, scaling and transformation was performed separately for the each fold.

Each of the preprocessing steps were taken on fold basis (training folds vs test fold) to ensure contamination of the learning process of the model was avoided.

In this section, the steps taken to split the datasets, handle missing values, scale the features, and transform the target variables as well as the rationale behind each step will be described. The The `K_Fold` class was designed and iteratively improved to perform these preprocessing tasks fold-wise, aimed to increase the predictions and reliability of model evaluation. This class is based on a custom implementation of k-fold cross-validation in a way that it allows hyperparameter tuning, training, and evaluating model in a loop. The method is designed to be applicable for both classification or regression algorithms, by making use of the continuous target values and numerical encoding of them as the task requires (more information and the evolution of the KFold Class is included in the project codes.)

The purpose of this approach is to ensure that no data from the validation set is used to influence the training of the model. The model is only allowed to learn from the training data, and any transformations applied to the validation data must be derived solely from the training set.

Imputation & Handling Missingness

If 30% of the feature is missing, it might be beneficial for a the column to be dropped. Dropping the features can be advantegous especially in the case of feature abundance. However, for the ccle dataset, some predictors were evaluated

valuable tumor-suppressors or oncogenes in cancers, for example: MAP3K1 (ENSG00000095015) which was recently reported to be involved in multiple cancers (Zheng et al., 2018). Therefore instead of being dropped column-wise, the values are imputed.

It is important to know that, imputation is essentially an artificial generation of data and can introduce biases. This places further the importance of proper feature selection strategies on downstream analysis.

For CCLE data, missingnes is assumed to be "Missing at Completely Random" (MAR) as the reason why for having missing data is not related to the missing columns itself. As many of the features showed skewed distribution, scikit-learn's `SimpleImputer(strategy = 'median' )` was selected to fill in missing values with the first median value.

Transformation and Scaling of the Features

As discussed in the earlier sections, many of the features exhibit clear divergence from normality, which violates assumptions of the certain machine learning models that expect normal distribution of data. To mitigte this issue, Yeo-Johnson transformation, which is a robust and flexible extension of the Box-Cox transformation, was applied. The formula for the Yeo-Johnson transformation is as follows (Yeo Johnson, 2000):

$$y(\lambda) = \begin{cases} \frac{(y+1)^{\lambda}-1}{\lambda}, & \text{if } y \geq 0 \\ -\frac{(-y+1)^{2-\lambda}-1}{2-\lambda}, & \text{if } y < 0 \end{cases} \tag{3}$$

Where:

- y is the original data point.
- λ is the transformation parameter (usually determined via maximum likelihood estimation (MLE)).

For $y \geq 0$, the transformation is similar to the Box-Cox transformation. For $y < 0$, the transformation uses a different formulation that accommodates negative values. For comparison, the definition of Box-Cox transformation is as follows (Aho, 2014):

$$y(\lambda) = \begin{cases} \frac{y^{\lambda}-1}{\lambda}, & \text{if } y > 0 \end{cases}$$

Where:

- y is the original data point, likewise Yeo-Johnson.
- λ is the transformation parameter.

Unlike Box-Cox transformation, Yeo-Johnson Transformation do not require datapoints to be strictly positive, and it accomodates to both positive and negative values of the data. This feature of Yeo-Johnson Transformation was essential on its selection, rendering it applicable to standardized gene expression matrices, which are composed of both positive and negative values.

However both, Yeo-Johnson and Box-Cox Transformations have a common challenge: **data leakage** when applying transformation across training and test sets. Using transformers globally, i.e. using the parameters derived from entire dataset, the model might inadverently learn information from the test set. This could lead to overfitting and inaccurate performance reports.

To address this issue, the functions of custom `K_fold` class ensured transformation was applied using parameters internal to each fold. Specifically, the transformation parameters are calculated and applied independently within each fold, in other words, transformation tailored and targeted one fold only. The dataset X is divided into k folds, where for each fold i, the Yeo-Johnson transformation parameters $\hat{\lambda}^{(i)}$ are computed using only the training data.

Another upside for Yeo-Johnson is, the scikit implementation accepts Standardization of data as an argument, which eliminates the need of using an extra standart scaler.

5 Model training, validation and testing

Next tasks involved training classifiers and regressors on CCLE mRNA data and testing their accuracy. For that, three classifiers were selected and the selection involved both non-parametric and parametric models for a throughout comparison. For regression, regularization based methods were selected in order to readily extract coefficients assigned for the features (table 4). RandomForestRegressor although was not included due to computational cost, might be compatible with this task as it is non-parametric and returns feature importances.

Algorithm	Parametric	Works with Null Values	Assumptions
K-Nearest Neighbors (KNN)	No	Yes (with imputation)	No Assumptions
Gaussian Naive Bayes (GNB)	Yes	No	Conditional Independence
Random Forest Classifier (RFC)	No	Yes (with imputation)	No Assumptions
Ridge Regression	Yes	No	Linearity, Homoscedasticity
Lasso Regression	Yes	No	Linearity, Homoscedasticity
Elastic Net Regression	Yes	No	Linearity, Homoscedasticity
Linear Regression	Yes	No	Linearity, Homoscedasticity

Table 4: Comparison of Classification and Regression Algorithms Selected for Comparison

5.1 Results of Classifier Performance Evaluation

For selected classifiers, the custom `KFold` class methods, were applied. This method returns the mean cross-validation scores, final test metrics (Accuracy Score, Precision Score, Recall Score and F1-Score), and the trained models.

Class	Mean Training Accuracy	Validation Accuracy	Precision	Recall	F1-Score	Test Accuracy
KNN	0.919028	0.435514	0.485008	0.435514	0.441822	0.517451
GNB	0.631883	0.487850	0.493664	0.487850	0.489819	0.582337
RFC	0.992722	0.482243	0.489515	0.482243	0.485020	0.553305

Table 5: Classification Model Performance Metrics: First three metrics, Training-Testing-Validation indicates Training Accuracy Mean across 5 folds, Testing Accuracy Mean across 5 folds and validation accuracy of the best performing model across cross-validation folds.

Each of the selected classification metrics points out a different aspect of the classification process. Precision, the ratio of true positive predictions to the total predicted positives, incidates how many of the predictive positives are actual positive instances. Recall, the ratio of true positive predictions to the total actual positive, indicates the abitility of the model to capture all relevant positive instances. Lastly, F1-Score, being the harmonic mean of the precision and recall, provides a single metric to evaluate model's performance in case of class imbalances.

Random Forest Classifier (RFC) performs perfectly on training set with an accuracy of more than 99%. This suggest overfitting, where the model memorizes the patterns in the training data but fails to generalize well during unseen data, which is also supported when the performance of the model drops by the half on testing and validation phases. Overfitting was intended to be avoided by limiting the maximum depth of the trees, however, more stronger constraints might be needed as the obseerved overfitting might be the result as the trees grow too complex and fit the noise of the data. To prevent this, constraints on the depth of trees can be further increased or minimum number of samples to split a node can be increased by passing higher values for `min_samples_split` argument to the model.

KNN appears performing well on training set however when tested on unseed data, it also fails to generalize well with only around 44% of accuracy. Upon nested cross-validation, the best performing model was returned having 10 neighbours, instead of 3 or 5 neighbours options given by the parameter grid. Therefore 'too many neighbours' might not be the major reason behind overfitting.

KNN relies on distances to the closest neighbours (e.g. Euclidean, Manhattan) between datapoints for classification and for high dimentional data, these distance metrices might not work well. In our case, the number of potentional combinations of values in a 494-dimensional space is extremely vast, data points are sparse while the KNN's ability to distinguish classes on proximity bases is reduced. Therefore, the curse of dimensionality might be the main result for suboptimal performance for KNN, as the distance between data points become less meaningful,in the sense that, the distances between points from the same class and points from different classes become similar with increasing number

of features. In order to overcome that, feature selection pre-KNN-classification, or dimensionality reduction via PCA or t-SNE (t-Distributed Stochastic Neighbor Embedding) might be applied.

Gaussian Naive Bayes performs the lowest in case of learning from the training set, which could be due to its independence assumption being violated by the CCLE dataset. GNB also expects features to follow a Gaussian Distribution however, this assumption also violates as the Shapiro-Wilk test has revealed majority of the features do not follow a Gaussian Distribution. This results points out that the GNB might not be an overall suitable model for data involving skewed features.

For each of those models, training, testing and validation accuracies are suboptimal, while precision and recall scores are close to each other, indicating the models have a balanced prediction performance classifying both positive and negative instances.

Suboptimal performance and overfitting encountered in classifiers might be due to the loss of information upon binning continuous target column into discrete categories. Even though quantile-binning instead of fixed-width binning was implemented, discretization might have resulted models to loss power to capture the relationship between the features and the target. If possible, binning of the target variable especially if there might be inherent relationships between features (e.g. genes involved in same pathways) present.

Best parameters found for each of the classifiers are:

Random Forest (RF): { max_depth = 5, min_samples_split = 5, n_estimators = 50 }
Gaussian Naive Bayes (GNB): { var_smoothing = 1 $\times 10^{-9}$}
K-Nearest Neighbors (KNN) : $\{$metric $=$ euclidean, n_neighbors $= 10$, weights $=$ distance$\}$

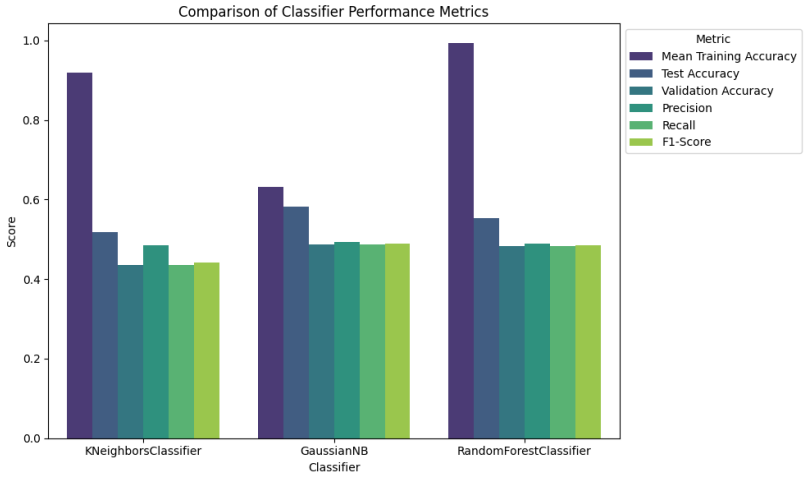

Figure 7: Comparison of classification performance across different models highlighting the trade-offs in model performance and generalization

5.2 Training Regressors on CCLE mRNA data and testing their accuracy:

Feature selection provides the basis for the second step of this project by identifying the most significant variables contributing to predictions. In order the facilitate the downstream feature selection process, regularization based methods, Ridge, Lasso and Elastic Net were selected for comparison. In order to see the effect of different classes of regularization, a simple linear model ordinary linear regression (Linear Regression) was also added the comparison. The cost functions for each model $\hat{\beta}$ to be minimized is given for each model where β_j is the j-th coefficient (or parameter) of the regression model, corresponding to the j-th feature x_j in the input vector $\mathbf{x}$. as the following:

Linear Regression (OLS): Minimizes the error without any regularization.

$$\hat{\beta} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \beta)^2 \right\}$$

Ridge Regression, L2 Regularization controls large coefficients, improving generalization in case of multicolinearity

$$\hat{\beta} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

Lasso Regression, L1 Regularization where l1 penalty can shrink coeffients to 0.0

$$\hat{\beta} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

ElasticNet Regression, combines L1 and L2 penalties.

$$\hat{\beta} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \beta)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2 \right\}$$

Regressor	Mean Training MSE	Mean Testing MSE	External MSE
Lasso Regression	0.0197	0.0207	0.0771
Ridge Regression	0.0024	0.0128	0.0733
ElasticNet Regression	0.0077	0.0129	0.0752
Linear Regression	0.0000	0.0209	0.0850

Table 6: Mean Mean Squared Error (MSE) for Training, Testing, and External Validation for Lasso, Ridge, ElasticNet, and Linear Regression (Ordinary Linear Regression). Mean train MSE is the average Mean Squared Error across the cross-validation folds for the training set. Mean Test MSE is the average Mean Squared Error across the cross-validation folds for the test set. External Test MSE it the Mean Squared Error of the tuned model returned to be validated on an external test set.

Linear regression (Ordinary Linear Regression) achieves a perfect training MSE with a test MSE of 0.0209 and an external MSE of 0.0805. This increasing trend suggest that Linear Regression model struggles with overfitting and shows poor generalization performance on unseen data, which is indicated by increasing MSE scores. Overfitting might be due to the fact that, the ordinary linear regression might assign large coeffients to each of the correlated features, leading those features to dominate the model. In the presence of multicolinearity, regularization methods might help prevent the correlated features to dominate the model.

For Lasso (L1 Regularization: forcing some coefficients to be 0), training mean MSE is higher compared to others, which suggest Lasso may not be fitting well to the training data in relation to ElasticNet and Ridge. Trained Lasso's performance becomes lower on testing fold and it drops even further when lasso is tuned and make predictions on the external validation set. This might be due to L1 regularization driving some coefficients to zero, leading to even more sparse model. In fact, it was observed that best Lasso model sets perhaps majority of the feature coefficients to zero, and due to this reason, this model class was not included in the downstream feature selection process.

Ridge (L2 Regularization: adding a penalty to the squared magnitude of coefficients) has the lowest training MSE, indicates that it fits the training data better than the others. The model also has lowest testing and validation errors, showing highest overall performance.

For ElasticNet, the mean training and testing MSE as well as the validation MSE, between the values for Lasso and Ridge. This reflects its combination of both L1 and L2 regularization, allowing for a balance between sparsity and coefficient shrinkage.

In conclusion, even though each of them model the relationship between features and target using linear equations, the regularization based linear models, which introduces penalties on the coefficients of the features outperform Ordinary Linear Regression in terms of generalization ability (lower external test MSE). This highlights, the role of regularization in preventation of overfitting. Ridge has the lowest external MSE, evaluated as the most robust model among the three regularized methods. This might be due to, even though the correlation values are calculated to be low (figure 6 and section 3.2 results), they are still impactful characteristics for both datasets, and Ridge performed better due to this spesific nature of the dataset.

On figure 8 the variation in performance across the folds was illustrated for the three regularization based models. Lasso Regression can be seen exhibiting higher training and testing MSE compared to Ridge and ElasticNet models.

For every model class, the 5-fold cross-validations returns the best performing models as the following:

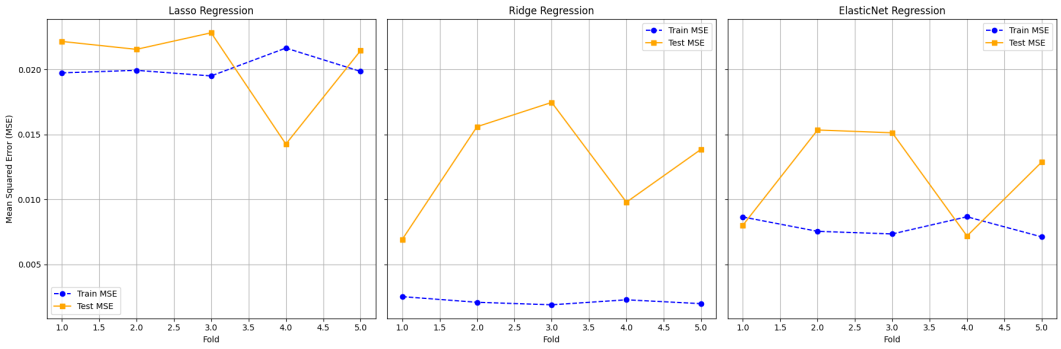

Figure 8: The performance of regularization based regression models; Lasso, Ridge, and ElasticNet, evaluated using five-fold cross-validation For each model, the training MSE (blue dashed line) and testing MSE (orange solid line) are plotted against the cross-validation fold number.

1. Ridge, with alpha=100.0 and solver='saga',
2. Lasso, with alpha=0.1.
3. ElasticNet, with hyperparameters alpha=0.1, l1_ratio=0.1, and selection='random'.

This results supported the selection of WRAP53 as one of the biomarker candidates, as it is the highest ranking feature according to best performing model (Ridge, with alpha=100.0 and solver='saga').

6 Comparison of Feature Selection Strategies

6.1 Feature Selection by the Best Model Coefficients

The feature importances extracted from coefficients used by the best (tuned and validated) Linear Regression, Ridge and ElasticNet Models were observed to be the following:

LR GeneIDs	Coefs	Ridge GeneIDs	Coefs	ElasticNet GeneIDs	Coefs
WRAP53	0.030507	WRAP53	0.011775	TXNDC11	0.011983
PIK3R5	0.021644	TXNDC11	0.011167	POLR3G	-0.014893
H3C7	0.020650	EIF4A3	0.009246	SEPTIN7	0.007306
EIF4A3	0.020952	CPE	0.009497	WTAP	0.009439
TXNDC11	0.019981	H3C7	0.008320	PDCD2	-0.007194
CPE	0.019492	EXOSC9	-0.010199	DCP2	-0.010026
POLR3G	-0.020098	SUV39H2	0.010375	GGCT	-0.009079
ELAVL1	-0.023990	SNRPD1	-0.008196	EXOSC9	-0.020434
METAP2	-0.021169	NUDT15	-0.008090	CAPRIN1	-0.008560
YIPF5	-0.031788	POLR3G	-0.013212	POLR3G	-0.014893

Table 7: Sorted Top 10 Most Important Features for Linear Regression, Ridge, and ElasticNet

WRAP53, WD repeat containing antisense to TP53 gene with EnsemblID:ENSG00000141499 was found to be included in most important 10 genes by Linear Regression and Ridge models.

6.2 Feature Selection by Sequential Feature Selection

This part is based on Scikit-Learn Documentation: Model-based and sequential feature selection

For comparison to the results obtained by model coefficients, each of the best performing models were passed to the scikit-learn's `SequentialFeatureSelector()`. For the selected features, gene names were fetched by Ensemble API using a custom function `get_gene_name()`. As patient data included gene names (symbols) as features, this step was necessary for compatibility from cell line expression matrices to the final patient gene expression matrix (`coad`).

Sequential Feature Selection (SFS) is a wrapper method that iteratively evaluates subset of features based on the model's performance and returns the most relevant features accordingly. Forward SFS, as it was used to select features in this

stage, starts with no features. At each step, it adds a feature that improves model's performance based on a scoring metric (which was selected as R2 in this stage). The iteration continues until a specified number of features are selected or no more features that dignificantly improve model's performance can be added.

Linear Regression Gene Names	Ridge Gene Names	ElasticNet Gene Names
MAP3K4	RFX3	CDH6
MRPS31	MAP3K4	GDF15
NKG7	MRPS31	CAMK4
CPSF6	GDF15	DEPTOR
TAS2R16	INTS15	PPIC
POLR1E	SETBP1	PTPRM
INTS15	CAMK4	ADGRG2
CAMK4	PPIC	KBTBD11
TPP1	TMEM45A	TMEM45A
RBP3	HSPA1B	HSPA1B

Table 8: Selected Gene Names from Linear Regression, Ridge, and ElasticNet Models using Sequential Feature Selection (SFS)

The table above shows the most important features found for each model class by SFS.

As coefficient based feature selection and SFS-based feature selection did ot match, the biomarker genes were selected based on biological and clinical relevance and validated on the basis of statical significance. As a result, the selected biomarkers were the following:

- **WRAP53 (p53 regulator of apoptosis and cell cycle):** is a regulator of p53 pathway. In 2020, p53 activation is reported to be suppressed by the irinotecan metabolite SN-38-induced cell damage in non-malignant epithelial colonic cells (Gunasegaran, Neilsen, & Smid, 2020).
- **MAP3K4 (Mitogen-Activated Protein Kinase Kinase Kinase 4)**: mutations in two other members of MAPK family was reported to lead to increased sensitivity to MEK inhibitors (e.g. Cobimetinib). This biomarket is also supported by the fact that Ras/Raf/MEK/ERK pathway are detected to have mutations in almost in 50% of colorectal cancer cases (Gong et al., 2018).
- **MRPS31 (Mitochondrial Ribosomal Protein S31)**: In 2020 a novel transcript lncMRPS31P5 was identified as a involved in colorectal cancer (Panza et al, 2020) . This gene can not be included as it was not a feature for Colon Adenocarcinoma data.

7 Clustering Colon Adenocarcinoma Patient Data on the Basis of Biomarkers

Both tables are evaluated on the basis of clinical and biological relevance to the genes to the reported molecular mechanisms underlying colon cancer and mechanism of action of the irinotecan.

Due to nature of data including high numbers of non-Gaussian distributed features (figures 1, 2 and 9) the Wilcoxon signed-rank test was selected as the statistical method to compare selected biomarkers with the reference gene.

The Wilcoxon signed-rank test is a non-parametric test to compare paired observations via evaluating whether the median difference between the paired samples is zero.

8 Clustering Power of two Biomarkers: WRAP53 and MAP3K4

First, the clustering power of two remaining biomarkers were statistically evaluated by Wilcoxon signed-rank test on two edge groups of CCLE and GDSC datasets: Resistant and Sensitive. Mild Response groups were excluded from the analysis.

The statistical analysis was performed using the Wilcoxon rank-sum test, yielding significant differences between the groups for both biomarkers: WRAP53 (with p = 0.00076) and MAP3K4 (with p value 1.29 x 10^{-7}) These results suggest that the expression levels of these genes may serve as potential indicators of irinotecan sensitivity.

Similarly, the Wilcoxon rank-sum test revealed significant differences between the groups for both biomarkers: WRAP53 (with p value 4.09 x 10^{-5}) and MAP3K4 (with p value 1.12 x 10^{-6}), supporting the hypothesis that those biomarkers could stratify patients based on their response to irinotecan treatment.

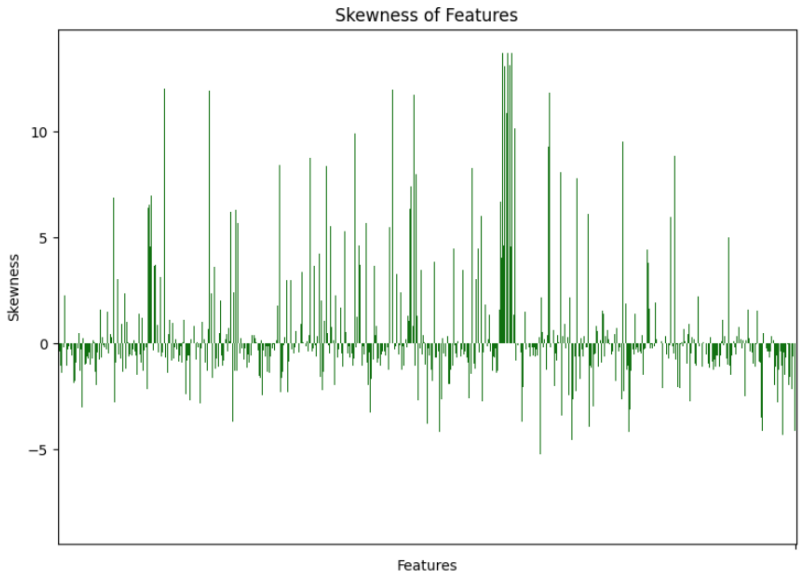

Figure 9: Features of Colon Adenocarcinoma dataset often shows significant level of positive skew, similar to CCLE and GDSC datasets.

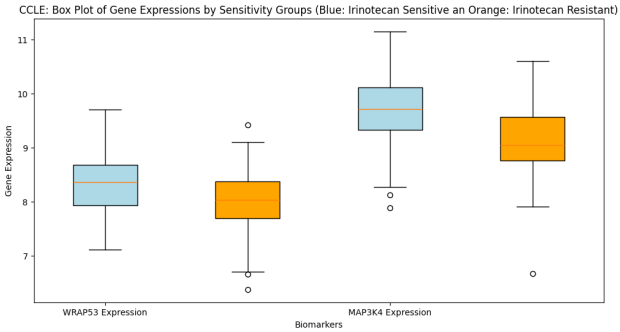

Figure 10: Box plots showing the expression levels of WRAP53 and MAP3K4 in two groups of colorectal cancer cell lines classified based on their sensitivity to irinotecan: sensitive (blue) and resistant (orange).

Upon observing these results, samples with low WRAP and MAP3K4 expression at the same time considered as Irinotecan Resistant, while samples with high WRAP53 and MAP4K3 expression considered as Irinotecan sensitive.

9 K-Means Clustering of Colon Adenosarcinoma (COAD) Patients to Identify Treatment Subgroups

For the second part of the task, K-Means Clustering was selected as the unsupervised learning method to subgroup patients, due to its computational efficiency, compatibility with PCA and allowence of using elbow method for finding the optimal numbers of clusters.

Colon Adenocarcinoma Patient Dataset (`coad`) did not include any null values. It has been standardized as K-Means clustering is sensitive to feature scaling, where features with larger ranges might dominate the clustering process.

In order to find most optimal K, the Elbow method is used. For this, first ineratia (sum of squared distances of samples to their nearest cluster center) for different K values are collected and and then plotted for corresponding number of clusters. The elbow point where inertia starts to decrease in a slower rate indicates the optimal number of clusters as it corresponds to the balance point between the number of clusters and explained variance.

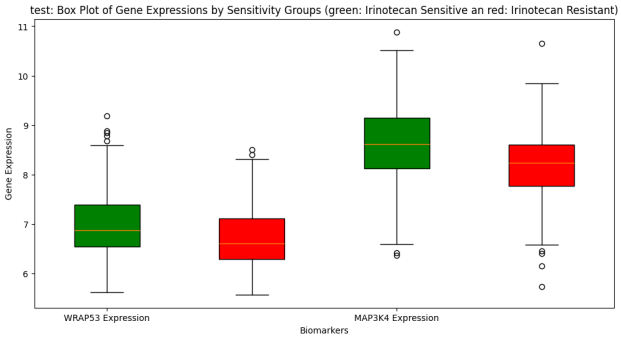

Figure 11: Box plots illustrating the expression levels of WRAP53 and MAP3K4 in two groups of colorectal cancer cell lines classified by their sensitivity to irinotecan: sensitive (green) and resistant (red) for the GDSC (test) set.

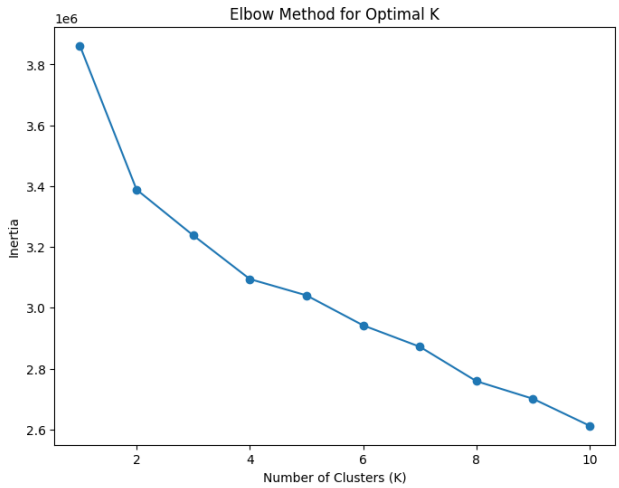

Figure 12: Elbow Method for Determining the Optimal Number of Clusters (K), where the 'elbow' point indicates the inertia decreases at a slower rate, and corresponds to the optimal number of clusters for the K-Means clustering algorithm.

In the case of COAD dataset, optimal K value obtained by the elbow method was $K = 2$ (figure 12).

For clustering, K-Means clustering was performed in combination with PCA, due to high dimensionality of dataset with more than 2000 features. In high dimensions, just like KNN, K-Means clustering might suffer from curse of dimensionality as well, where all points appear equidistant. Number of clusters = 2 was passed to the K-Means as an argument and PCA results were plotted as a scatterplot (figure 13).

To assess whether biomarker gene expression have significant differences between two patient clusters obtained by K-Means clustering algorithm, Wilcoxon rank-sum test was performed. This test is non-parametric therefore compatible with non-Gaussian feature distribution along COAD data (figure 9). The results yield a p-value, which indicates whether the differences in expression between the two clusters are statistically significant, indicated by values of $p < 0.05$.

Statistical significance assessed by the Wilcoxon test showed significant difference in WRAP53 expression between clusters ($p = 1.73 \times 10^{-6}$), while there was no significant difference in MAP3K4 expression ($p = 0.78$). This result indicates indicating clusters have distinct expression profiles for WRAP53, but not for MAP3K4.

Figure 13: The scatterplot shows the distribution of the samples in the first two principal component axes and the points are colored based on the cluster assignment.

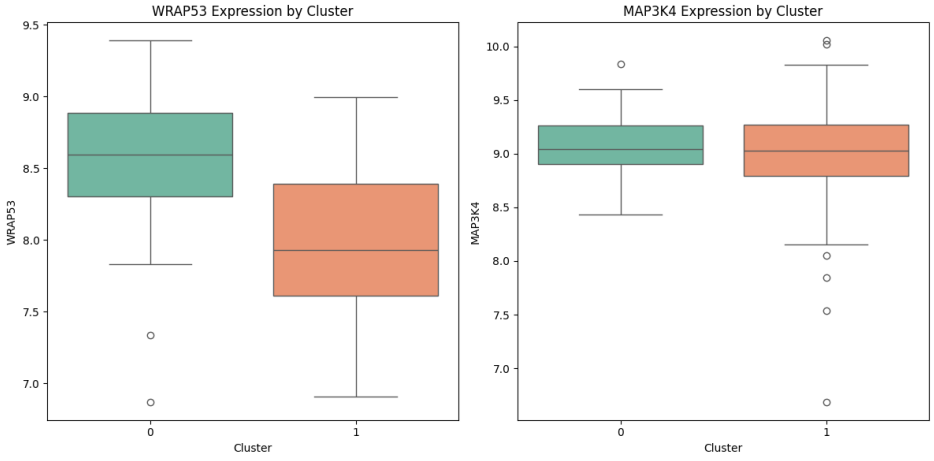

Figure 14: Boxplots of WRAP53 and MAP3K4 expression across two K-means clusters. The boxplots display the standardized expression levels of WRAP53 (left) and MAP3K4 (right) for two clusters. Clusters labeled with 0 are higher expression clusters and 1 is lower expression clusters for candidate biomarkers.

10 Conclusion:

Irinotecan sensitive cancer cell lines were found to express higher levels of a set of candidate biomarker genes, including WRAP53 and MAP3K4, with a statistically significant difference in comparison with irinotecan resistant group.

Then, patient subgroups obtained by unsupervised learning algorithms are compared. The comparison focused on how the expression patterns of WRAP53 and MAP3K4 in the identified patient clusters align with those in cell lines.

As a result, significant difference in WRAP53 expression between the two clusters ($p = 1.73 \times 10^{-6}$), with higher expression in Cluster 0. As WRAP53 was found to have an higher expression in irinotecan sensitive cell lines, this cluster may respond better to the drug.

For the case of MAP3K4, no significant difference was found between patient clusters was found ($p = 0.78$). This suggests that, even though it was an significant factor to distinguish between drug-resistant and drug -sensitive cell line models (figure 10 and 11), it might not be a reliable biomarker for distinguishing between patient subgroups (figure 14).

This results indicated the gap between cell line models and human-patient data. Biomarkers that are discovered across cell lines ultimately needs to be validated on patient datasets to confirm their predictive value.

For a future direction, it could be checked whether the patients in cluster 0 which indicated sensitivity to irinotecan based on higher expression levels of biomarker gene WRAP53 shows better prognosis under treatment.

11 References

- Aho, K. A. (2014). *Foundational and Applied Statistics for Biologists* (First ed.). Chapman Hall / CRC Press. ISBN 978-1439873380.
- Boehm, B. (n.d.). Yeo-Johnson transformation. Retrieved from `https://bradleyboehmke.github.io/HOML/engineering.html`
- Ghandi, M., Huang, F. W., Jané-Valbuena, J., Kryukov, G. V., Lo, C. C., McDonald, E. R. 3rd, Barretina, J., Gelfand, E. T., Bielski, C. M., Li, H., Hu, K., Andreev-Drakhlin, A. Y., Kim, J., Hess, J. M., Haas, B. J., Aguet, F., Weir, B. A., Rothberg, M. V., Paolella, B. R., Lawrence, M. S., Akbani, R., Lu, Y., Tiv, H. L., Gokhale, P. C., de Weck, A., Mansour, A. A., Oh, C., Shih, J., Hadi, K., Rosen, Y., Bistline, J., Venkatesan, K., Reddy, A., Sonkin, D., Liu, M., Lehar, J., Korn, J. M., Porter, D. A., Jones, M. D., Golji, J., Caponigro, G., Taylor, J. E., Dunning, C. M., Creech, A. L., Warren, A. C., McFarland, J. M., Zamanighomi, M., Kauffmann, A., Stransky, N., Imielinski, M., Maruvka, Y. E., Cherniack, A. D., Tsherniak, A., Vazquez, F., Jaffe, J. D., Lane, A. A., Weinstock, D. M., Johannessen, C. M., Morrissey, M. P., Stegmeier, F., Schlegel, R., Hahn, W. C., Getz, G., Mills, G. B., Boehm, J. S., Golub, T. R., Garraway, L. A. (2019). Next-generation characterization of the Cancer Cell Line Encyclopedia. Nature, 569(7757), 503–508. https://doi.org/10.1038/s41586-019-1186-3
- Espín-Pérez, A., Portier, C., Chadeau-Hyam, M., van Veldhoven, K., Kleinjans, J. C. S., de Kok, T. M. C. M. (2018). Comparison of statistical methods and the use of quality control samples for batch effect correction in human transcriptome data. PLOS ONE, 13(8), e0202947. https://doi.org/10.1371/journal.pone.0202947
- Xu, Y., Villalona-Calero, M. A. (2002). Irinotecan: mechanisms of tumor resistance and novel strategies for modulating its activity. Annals of Oncology, 13(12), 1841–1851. https://doi.org/10.1093/annonc/mdf337
- Armand, J. P., Ducreux, M., Mahjoubi, M., Abigerges, D., Bugat, R., Chabot, G., Herait, P., de Forni, M., Rougier, P. (1995). CPT-11 (irinotecan) in the treatment of colorectal cancer. *European Journal of Cancer*, 31A(7-8), 1283–1287. https://doi.org/10.1016/0959-8049(95)00212-2
- Gong, S., Xu, D., Zhu, J., Zou, F., Peng, R. (2018). Efficacy of the MEK inhibitor Cobimetinib and its potential application to colorectal cancer cells. *Cell Physiology and Biochemistry*, 47(2), 680–693. https://doi.org/10.1159/000490022
- Gunasegaran, B., Neilsen, P. M., Smid, S. D. (2020). P53 activation suppresses irinotecan metabolite SN-38-induced cell damage in non-malignant but not malignant epithelial colonic cells. Toxicology in Vitro, 67, 104908. https://doi.org/10.1016/j.tiv.2020.104908
- Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media.
- Shapiro, S. S., & Wilk, M. B. (1965). "An analysis of variance test for normality (complete samples)." *Biometrika*, 52(3–4), 591–611. 10.1093/biomet/52.3-4.591. JSTOR 2333709. MR 0205384. p. 593.
- Yeo, I., & Johnson, R. A. (2000). A New Family of Power Transformations to Improve Normality or Symmetry. Biometrika, 87(4), 954–959. https://doi.org/10.1093/biomet/87.4.954
- Nusinow, D. P., Szpyt, J., Ghandi, M., & others. (2020). Quantitative proteomics of the Cancer Cell Line Encyclopedia. *Nature*, 579, 483–489. https://doi.org/10.1038/s41586-019-1413-6
- Yang, W., Soares, J., Greninger, P., & others. (2013). Genomics of Drug Sensitivity in Cancer (GDSC): A resource for therapeutic biomarker discovery. *Nature*, 505, 585–590. https://doi.org/10.1038/nature12966
- James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). An Introduction to Statistical Learning. UC Berkeley Statistics Department. Retrieved from `https://www.stat.berkeley.edu/ISLR_First_Printing`