THE COOPER UNION FOR THE ADVANCEMENT OF SCIENCE AND ART
ALBERT NERKEN SCHOOL OF ENGINEERING

Pancancer Analysis to Bridge the Gap between Metabolomics and Transcriptomics through Machine Learning

By

Junbum Kim

A thesis submitted in partial fulfillment of the requirements for the degree of
Master of Engineering

Advisors

Sam Keene and Ed Reznik

THE COOPER UNION

FOR THE ADVANCEMENT OF SCIENCE AND ART

ALBERT NERKEN SCHOOL OF ENGINEERING

This thesis was prepared under the direction of the Candidate's Thesis Advisor and has received approval. It was submitted to the Dean of the School of Engineering and the full Faculty, and was approved as partial fulfillment of the requirements for the degree of Master of Engineering.

____

Dean, School of Engineering Date

____

Professor Sam Keene Date

Thesis Advisor

Acknowledgement

I would like to thank Sam Keene for his inspiration, faith in my efforts, and determination to see me pursue studies in machine learning.

I would like to thank Ed Reznik for his continuing guidance through my research in a rather unfamiliar domain of computational biology.

I would like to thank Eric Liu for his professional and personal advice in the pursuit of research and publication.

I would like to that Renzo DiNatale for his availability and willingness to help in my academic pursuits in a rather unfamiliar domain of computational biology.

I would like to thank Jan Krumsiek and Elisa Benedetti for providing the metabolomics and genomics data, and knowledge in the domain of multiomics.

I would like to thank my parents for their endless support for my education and wishing me best of luck during my hardest times. Without all these people, I would not have been able to finish this paper.

Abstract

In this paper, we present the brand-new concept of in silico metabolomics. The idea is that metabolomics data can be imputed through computational methods from already available RNA expression data. We show that LASSO regression can effectively predict metabolomics using transcriptomics data under sparsity condition given that the relationship is linear. We validate the importance of this prediction by demonstrating the importance of metabolomics data to immune profiling, and thus potentially to immunotherapy. We first analyze kidney cancer data and generalize our findings to pan-cancer data, including breast cancer, colon cancer, liver cancer, ovarian cancer, prostate cancer, and lymphoma to identify previously unknown potential relationships between metabolites and genes.

Table of Contents

List of Figures

Introduction

Genomics and transcriptomics are one of the most wildly studied topics in bioinformatics to date. As they play an important role on cellular reproduction, genes and transcriptomes are known to encode all necessary biological instruction and information to form a human. The richness of the dataset is useful when it comes to cancer treatment especially given that genomic sequencing yields us expression levels of 30,000 different genes in the microbiome. For instance, genomic profiling of a cancer sample could tell us specific information about cancer beyond typical types. It is used not only to classify whether a sample is normal or tumor, but also whether a kidney cancer is clear-cell renal cell carcinoma (ccRCC) or chromophobe renal cell carcinoma, a rarer type. Genomics and transcriptomics data are one of the crucial pieces of information that is collected and analyzed for cancer patients. For example, transcriptomics is often used to profile immune signatures through a pipeline called single sample Gene Signature Enrichment Analysis (ssGSEA). ssGSEA imputes immune profile based on transcriptomics data. This profile ultimately helps the decision making of individualized targeted immune treatment.

Given the background, we would like to focus on proposing two arguments in this paper: first, to show the potentials of using metabolomics data to enrich information about potentially targetable immune reactions; second, to prove metabolomics data is reasonably imputable with transcriptomics data. In doing so, we would like to highlight the importance of tying metabolomic analysis to immune signatures to reveal the underlying metabolomic mechanisms for immunosuppression. We would also like to demonstrate that we can potentially collect metabolomic data without mass spectrometry by using transcriptomics data, which are already available.

To demonstrate the feasibility of the two arguments, we will use the metabolomics data to predict immune signatures and use transcriptomic data to predict metabolite levels in this paper. We will demonstrate that this analysis is possible mainly through ccRCC samples, but will show that is possible for all types of cancer including breast cancer (BrCa), colon cancer (COAD), lymph cancer (DLBCL), liver cancer (HCC), ovarian cancer (OV), and prostate cancer (PRAD). We will use different techniques ranging from univariate correlational analysis, to more complicated models like LASSO regressors and random forest regressors to identify which metabolites and immune signatures are predictable and which metabolites and genes are important in making such decisions.

Table of Nomenclature

Keyword: Pancancer, Multiomics, Machine Learning

1. Pancancer Analysis
    a. BrCa – Breast Cancer
    b. COAD – Colon Adenocarcinoma (Colon Cancer)
    c. DLBCL - Diffuse large B-cell lymphoma
    d. HCC – Hepatocellular Carcinoma (Liver Cancer)
    e. OV – Ovarian Cancer
    f. PRAD – Prostate Adenocarcinoma
    g. ccRCC – Clear Cell Renal Cell Carcinoma
2. Multiomics
    a. Metabolomics – Study of metabolites within a batch cell
    b. Genomics – Study of which Genes exist in DNA
    c. Transcriptomics – Study of the expression level of genes in RNA in the process of transcription
3. Immunotherapy
    a. Immunosuppression – Indication of low immune activity
    b. Immunoinhibition – Indication of high immune activity
4. Data Collection and Processing
    a. Mass Spectrometry – Separation of substance through its chemical properties
    b. RNA-seq – Technology to sequence Genes and Transcriptomes from cells

Background Study

1. Cancer treatment through immunotherapy

There are various practices to cure, shrink or stop the progression of a cancer. Some of the forms include, surgery, chemotherapy, radiation therapy, immunotherapy, hormone therapy, targeted drug therapy, clinical trials and so forth. Among these, the hot potato now is immunotherapy. Immunotherapy tries to educate the immune system to recognize and attack specific cancer cells, boost immune cells to help them eliminate cancer. The characteristics are that immunotherapy can work on many different types of cancer in addition to traditional forms of treatments, may be effective at preventing recurrence, and that it has different side effects compared to other treatments.

Immunotherapy enables the immune system to recognize and target cancer cell, thus could be effective to multiple cancer types at the same time. There are 21 different cancer types that are being treated using immunotherapy including bladder cancer, brain cancer, breast cancer, cervical cancer, childhood cancer, colorectal cancer, esophageal cancer, head and neck cancer, kidney cancer, leukemia, liver cancer, lung cancer, lymphoma, melanoma, multiple myeloma, ovarian cancer, pancreatic cancer, prostate cancer, sarcoma, skin cancer, and stomach cancer. Immunotherapy has been an effective treatment for patients that have been resistant to chemotherapy and radiation treatment.

Cancer immunotherapy offers the prevention of long-term cancer remission, because immune system can remember cancer cells. Clinical studies on long-term overall survival have shown that the beneficial responses to cancer immunotherapy treatment are durable – that is, they may be maintained even after treatment is completed.

Immunotherapy has different side effects from chemotherapy and radiation. Since immunotherapy is focused on the immune system, it has different side effects. The side effects vary according to each therapy and how it interacts with the body. Conventional cancer treatment like chemical or radiological therapy include hair loss and nausea. Potential side effects to an overstimulation or misdirection of immune system may range from minor symptoms of inflammation to major conditions like autoimmune disorders.

2. Multi-omics analysis

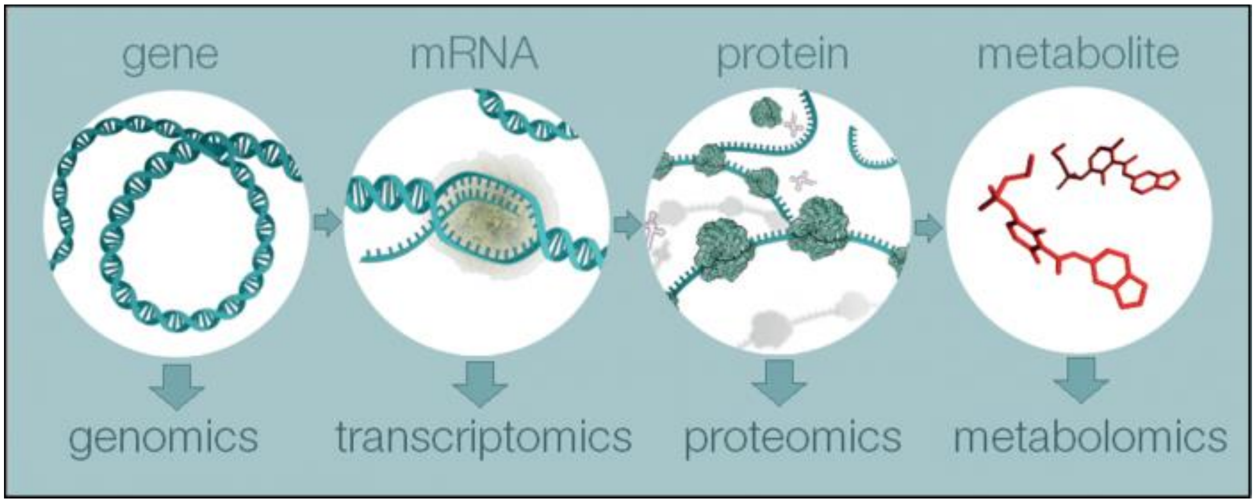

Figure 1. Category of common fields in Multiomics

There are largely four divisions in the study of multi-omics: genomics, transcriptomics, proteomics, and metabolomics. Genomics is the study of DNA and genetic information within a cell, transcriptomics is the study of RNA and differences in mRNA expression, metabolomics is the study of substrates and products of metabolism which are influenced by both genetic and environmental factors. Examples of metabolites include glucose (a sugar), glutamine (an amino acid), cholesterol (a lipid), and stearic acid (a fatty acid).

DNA forms a double helix shape, where each of the helix forms a helix during transcription. Genes are sequence of base pairs located within the DNA or RNA. Most genes are the same in all people, but a small number of gene (less than 1 percent of the total) are slightly different between people. Alleles are forms of the same gene with small mutations in their sequence of RNA bases. These small differences contribute to each person's unique physical features. Transcriptomics is a quantification of how much each of the genes are expressed or inhibited during transcription, a process in cell replication to form two DNAs from one.

The reason why this multiomics study works is because we go from a dataset with particularly rich feature set to a fewer one. For instance, the usual number of genes recorded are in the range of tens of thousands, whereas the number of metabolites measured almost never exceeds a thousand. Also, the reason why it is important to study metabolomics tied with genomics is because a lot of different disease share these genomic and metabolomic pathways. In this study we are will be dealing with three feature sets in total, one with metabolomics, one with transcriptomics, and one with immune profiles imputed from the genomics data.

3. Measuring level of Metabolites

Mass spectrometry either through liquid or gas chromatography and nuclear magnetic resonance are the two ways to measure levels of metabolite. The most common practice is to collect data through mass spectrometry. The basic principle behind the technology is that there is a given boiling point and mass to charge ratio for all biological compounds and we could use these properties to identify them. In gas chromatography, metabolites are boiled and accelerated through a weak electric field through a long chamber filled with inertial gas. The time it takes for each metabolite to reach the other end of the chamber varies as each of them has different boiling points and mass to charge ratio. In the entire process inertial gas act as a solvent to hinder the progression of solutes to increase the spatial separation between the particles. At the end, the intensity of particle coming out on the other end of the pipe is measured and recorded as in the Figure 2. Each of the peaks in the figure identify a metabolite.

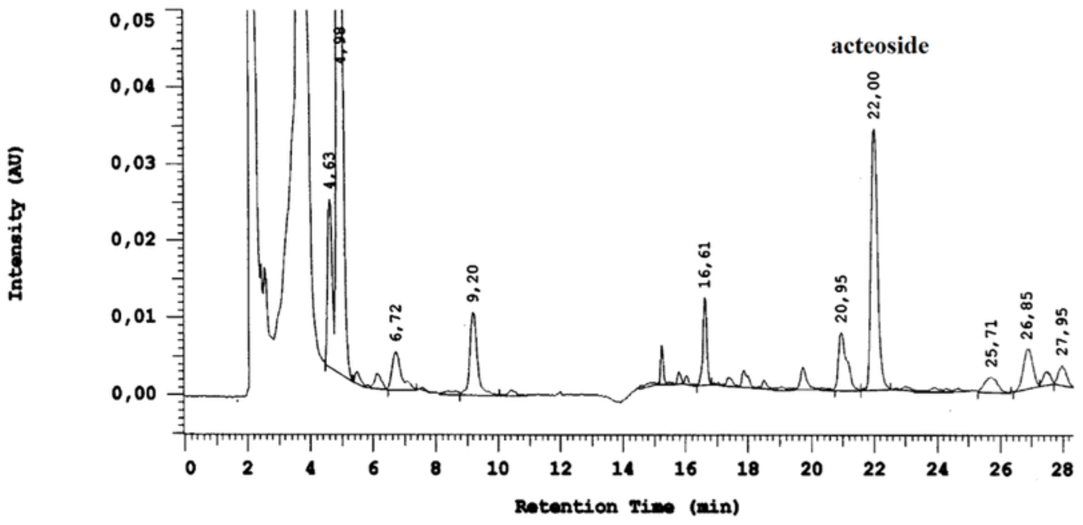

Figure 2. Sample Chromatogram

4. Measuring Genomics and Transcriptomics

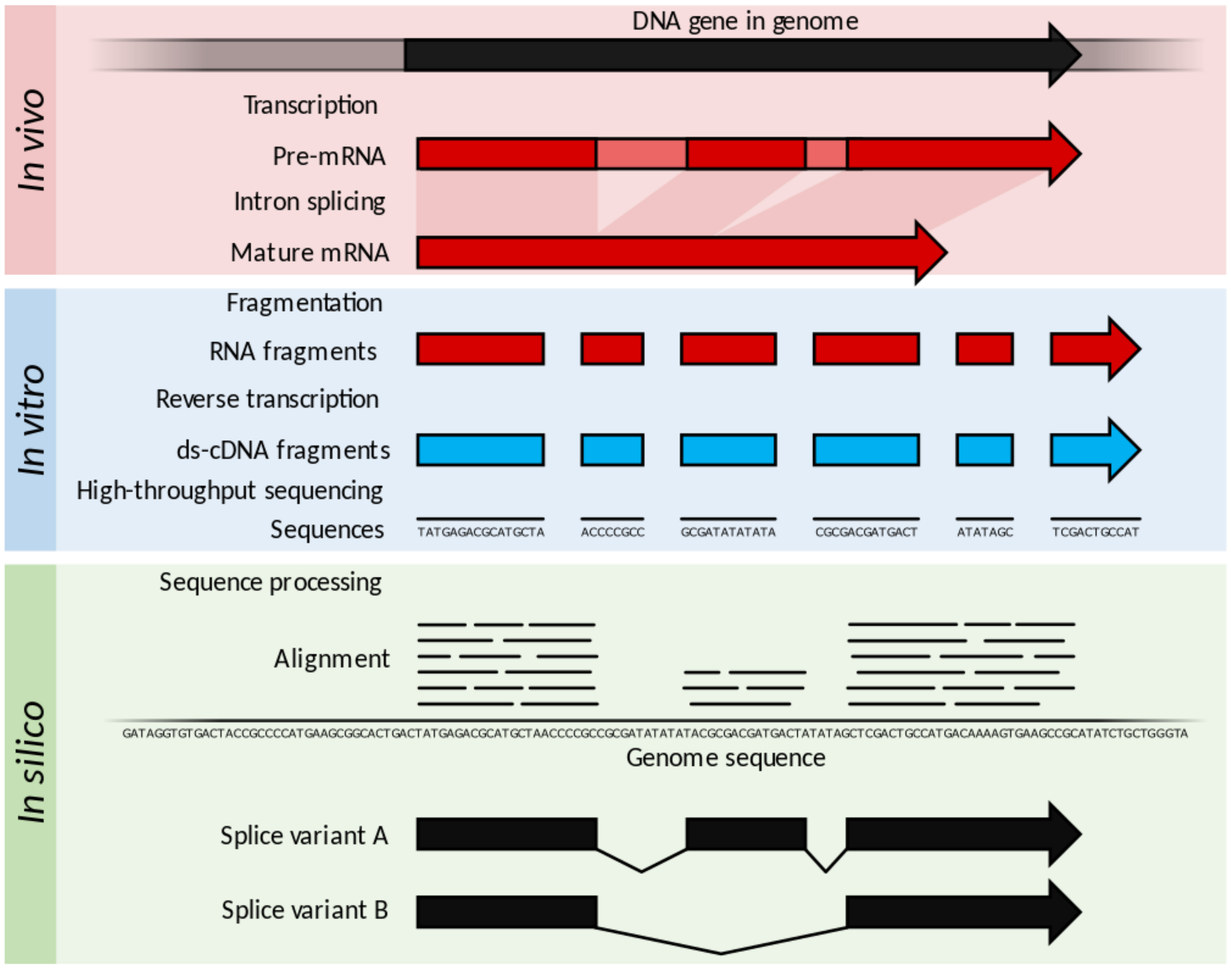

Figure 3. RNA-seq processing pipelin

The modern technology that allows us to seamlessly gather gene expression level is called RNA-seq. The modern-day RNA-seq is done largely in three steps: in vivo, in vitro, and in silico. In vivo means within living things, in vitro inside a glass, and in silico through computation.

In vivo, DNA undergoes transcription, where the double helix shape unzips into pairs of helices. After this stage, the RNA undergoes intron splicing and becomes a mature mRNA, messenger RNA. In vitro, these mRNAs are fragmented into smaller segments.

These mRNA fragments are reverse transcribed into ds-cDNA fragments and measured through high-throughput sequencing. The resulting fragments of sequence are collected in text form. In silico, the fragments of sequences are aligned to a known reference genome through dynamic programming. This data can be annotated to where expressed genes are and what their relative expression levels are.

5. What is special about this data?

Most metabolomics research is done on a few specific target metabolites. This study was an untargeted metabolomics experiment as opposed to targeted metabolomics. Instead of looking to validate a specific biochemical hypothesis, we were looking for potentially targetable pathways for immunotherapy. One advantage of untargeted metabolomics study is that we have access to the full picture of the metabolomic interaction within the microenvironment.

We also have data over multiple studies and across different cancer types. So we can compare and contrast metabolomic behaviors across different cancer types, and potentially identify shared metabolomic pathways between two different cancer types. One of the down sides of having such dataset is that we must deal with different batch effects, but the upside of it is that any discovery of correlation across these datasets more firmly validates a correlation.

6. Table of Datasets

Dataset ID	Type of Cancer	# Total samples	# Normal Samples	# Tumor Samples	# Metabolomic Features	# Genomic Features	# Immune Features
BRCA_Tang	Breast Cancer	18	0	18	356	23094	209
BRCA_Terunuma	Breast Cancer	108	47	61	259	20067	209
COAD	Colon Adenocarcinoma	76	39	37	111	22578	209
DLBCL	Diffuse Large B Cell Lymphoma	62	0	62	448	22271	209
HCC	Hurthle Cell Carcinoma	28	0	28	657	22357	209
OV	Ovarian Cancer	45	0	45	374	20988	209
PAAD (PDAC)	Pancreatic Ductal Adenocarcinoma	39	12	27	258	20067	209
PRAD	Prostate Adenocarcinoma	137	46	91	117	28018	209
RC12	Clear Cell Renal Cell Carcinoma	32	0	32	367	21461	209
RC18	Clear Cell Renal Cell Carcinoma	136	47	89	665	[22463, 23033][1]	209

Table 1. Statistics of collected data

[1] First number in the bracket is for RC18 Flow cohort data and second for RC18 Multiregional dataset

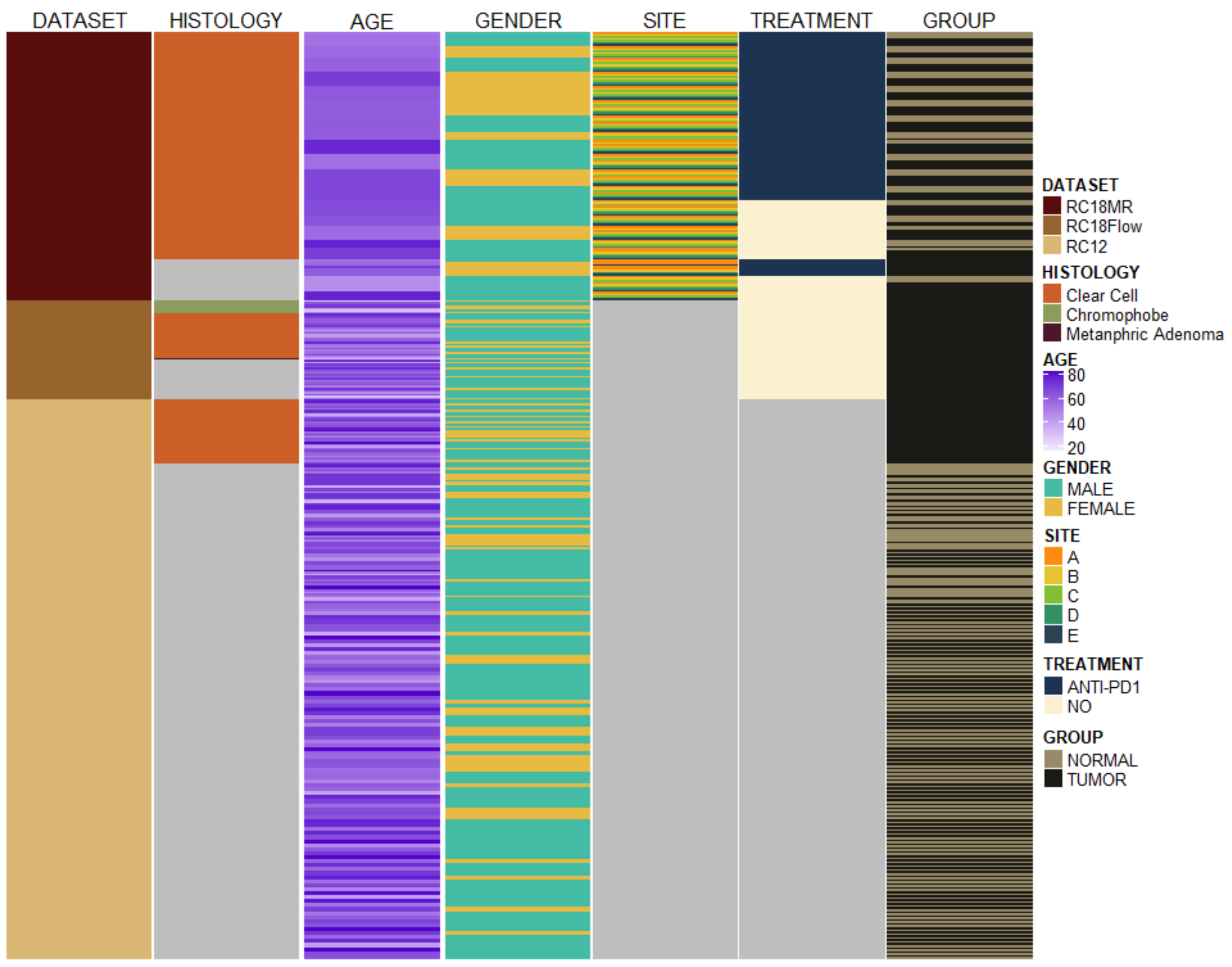

Figure 4. Demographic information about the data

Figure 4 is a visualization of the demographic information of the main dataset we will be dealing with. The kidney cancer dataset primarily consists of three datasets, RC12, RC18 multiregional, and RC18 flow cohort dataset. Each of the datasets come from different places and records different features. The number behind the datasets refer to which year the experiment was done on, and since these involve mechanical procedures to collect the data, each dataset has distinct batch effects. Whereas RC12 dataset is a retrospective sample, RC18 multiregional dataset was collected planning forward for this experiment and hence is one of the most feature rich dataset. RC18 multiregional data contains samples from multiple regions of kidney: A was the top part, E was the bottom part, and B, C, D were sides of the kidney. It also includes information about whether the patient was exposed to certain treatments.

Machine Learning Model Architecture

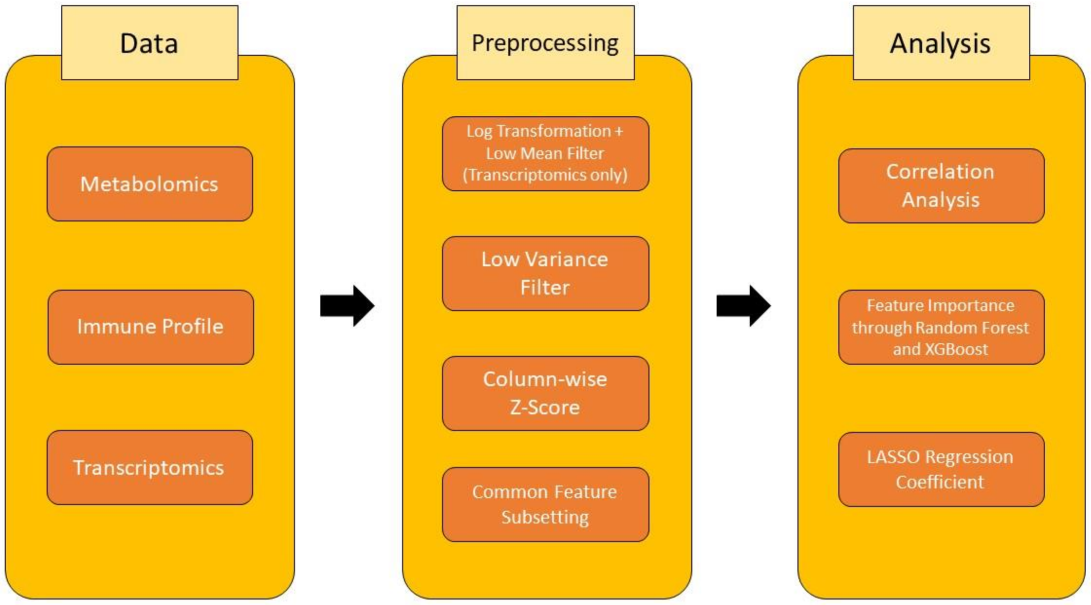
Data
Metabolomics
Immune Profile
Transcriptomics
Preprocessing
Log Transformation + Low Mean Filter (Transcriptomics only)
Low Variance Filter
Column-wise Z-Score
Common Feature Subsetting
Analysis
Correlation Analysis
Feature Importance through Random Forest and XGBoost
LASSO Regression Coefficient

Figure 5. Overall Data Processing Pipeline

There are three algorithms that we ended up for model selection. It is important to note that the task at hand is not to get the highest accuracy, but to figure out which metabolites and genes is important to the immune system. Hence, the first model of choice was random forest, which are great at providing a smooth gradient of imputed feature importance. The second model that we used was the LASSO model, which are advantageous in high dimensional regression where there the number of samples is limited. The LASSO provides a simple yet powerful and interpretable tool for the data. The last algorithm we used in this experiment was a spearman rank correlation test evaluating the correlation and p-value, the false discovery rate, for the experiment. Preprocessing stage will be dealt in more detail in a later section.

Exploratory Data Analysis

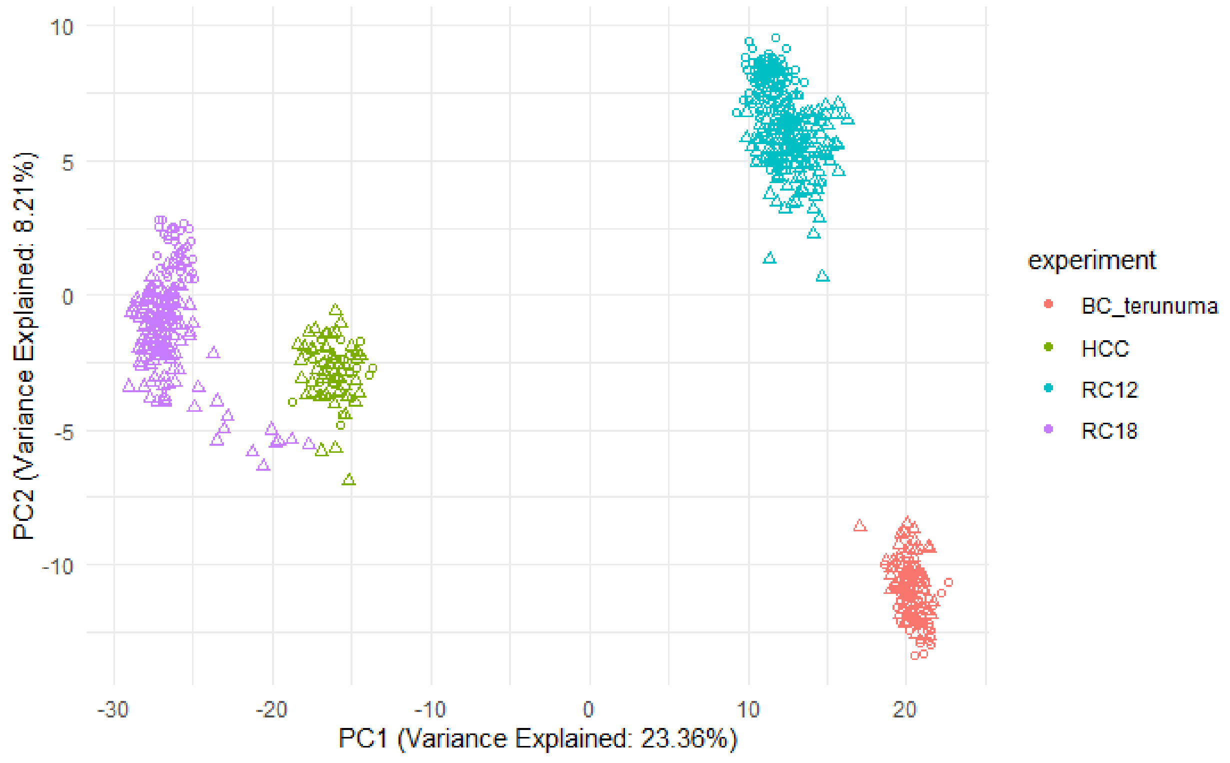

Figure 6. Heterogeneity in the dataset

With Figure 1Figure 6, the first thing we discovered was that there were heavy batch effects with the metabolite levels. The figure above shows the Principal Component Analysis (PCA) on the data. The first principle component axis separated data into four distinct clusters that indicated each of the experiments. This indicated that either the dataset required some form of normalization or cross-comparison was not possible with the scale of data at hand. The second principle axis separated normal and tumor samples for breast cancer, hepatocellular carcinoma, and renal cell carcinoma. Circle and triangle each denoted normal and tumor cells.

PCA Projection of ccRCC multicohort dataset

The first part of analysis tries to predict immune markers from metabolite levels. For the sake of simplicity, we start our analysis with RC18 multicohort dataset. The dataset includes 114 kidney cell data across with 67 tumor samples. The following is the first two axis of the Principle Component Analysis projection of the metabolomic data, color coded with immune score from the immune data.

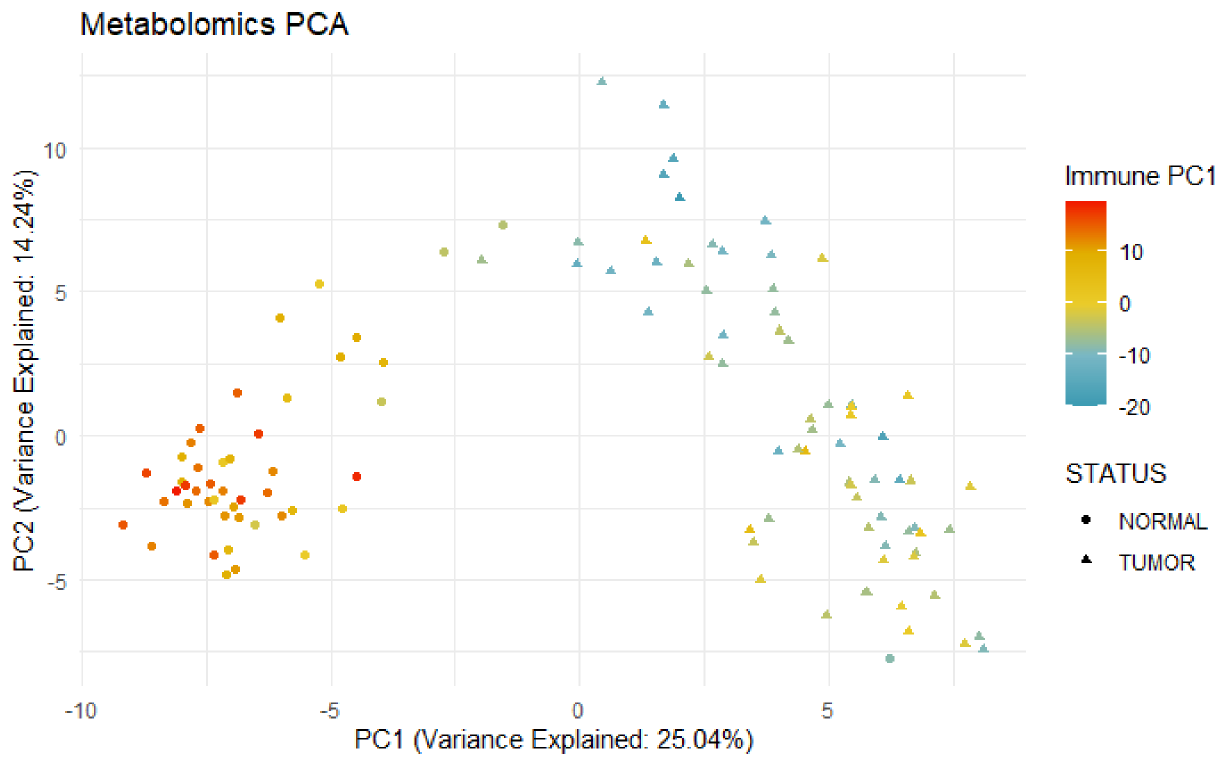

Figure 7. PCA projection of tumor and normal samples of the RC18 data

Figure 7 demonstrates formation of two distinct clusters: one on the left and one on the right. The cluster on the left are normal samples and on the right are tumor samples. This firstly demonstrates that the metabolite level between normal samples and tumor samples are easily linearly separable. More importantly, we see a clear distinction in the immune score between the two clusters. The tumor samples are marked with high immune activity compared to the normal samples.

Data Preprocessing

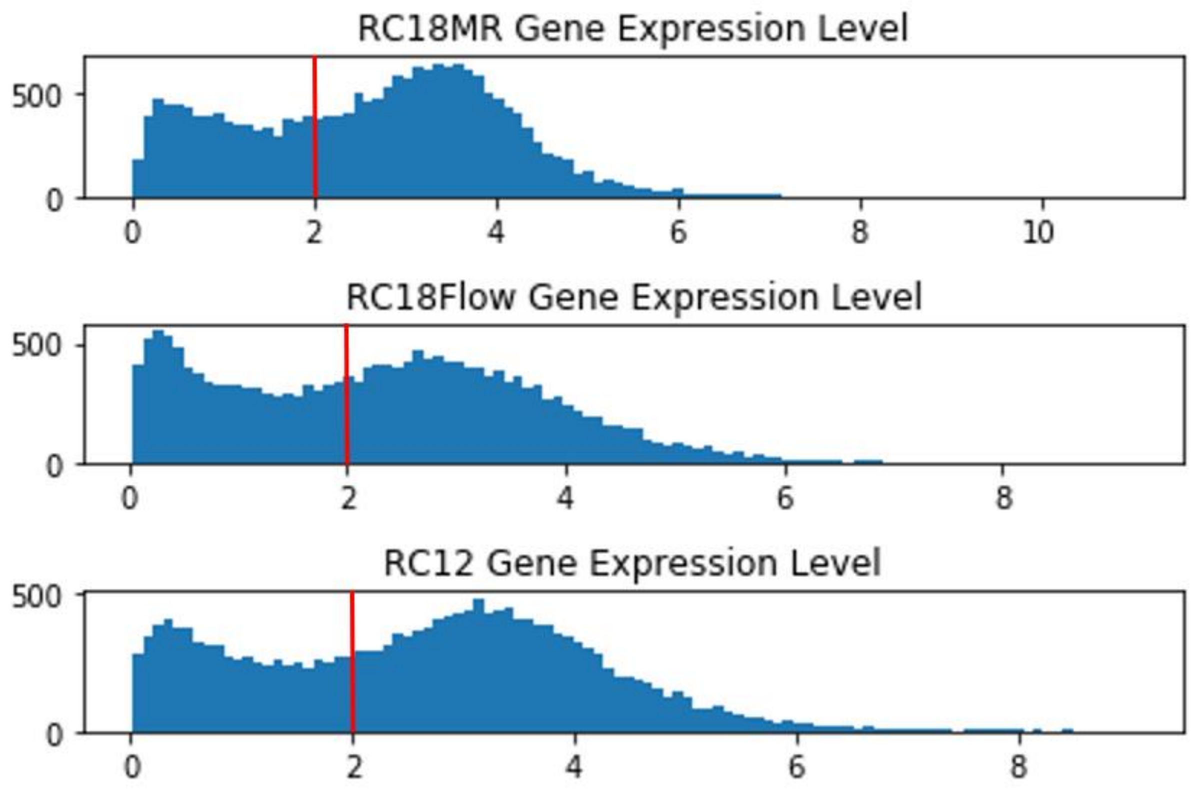

Figure 8. Gene Expression Levels in Different Cohorts

1. Filters

a. Low Mean Filter for log Gene Expression Level

b. Low Variance Filter for all datasets (Metabolomics, Immune, and Transcriptomics)

We started analysis with the kidney cancer dataset. We took the tumor samples in the RC12, RC18 flow cohort, and RC18 multi cohort data and filtered all features through a set intersection function to only consider features that were common across all cohorts. We applied log transformation: $x' = \log{(1 + x)}$ to evenly distribute datapoints on the positive respective axis, as collected raw data came in an exponential scale. We first filtered add genes that had lower than 2 log expression level as we considered them not expressed enough and are signals that yield low SNR. On top of that we applied low variance filter to filter out features that had lower than 0.1 variance in this transformed domain as features with low variance provide less information of interest and are more susceptible to noise. For transcriptomics data specifically, we applied a low mean filter to filter out genes that did not have a desired mean expression level.

2. Normalization

a. Standard Transformation (z-score)
b. Mean Matching
c. Quantile Transformation
d. Power Transformation

The last step after was to apply normalization on the different dataset to remove batch effects and harmonize them. We explored various methods including standard transformation, pure mean matching, quantile transformation, and power transformation as an effort to transform distributions into comparable gaussian distributions but none of the above improved the performance of the overall model. Hence, we decided to stick with z-scoring which is the most common form of transformation.

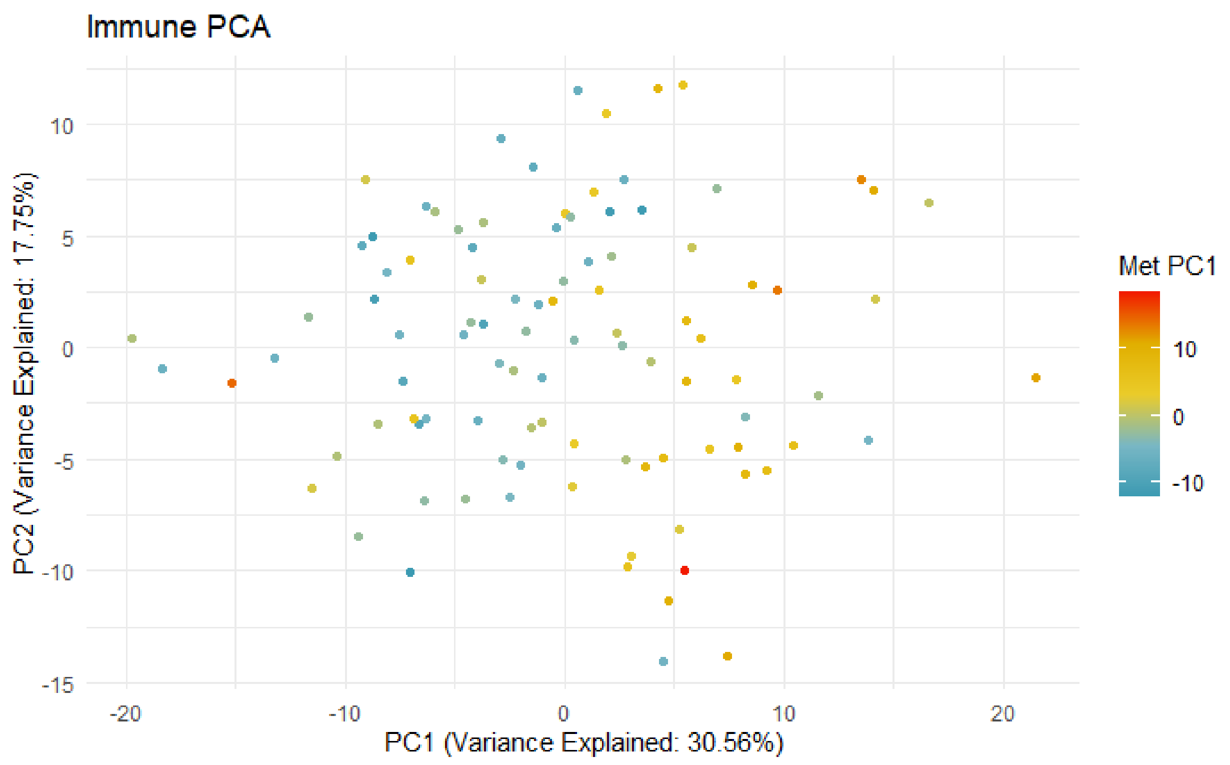

Figure 9. Principal Component Analysis on Immune Features

As a quality assurance check for the dataset, we analyze the relationship between metabolomics and immune features through PCA. From the plot, the immediately observable result is that the first two principal component of the immune feature describes a huge proportion of the variance of the dataset. Alongside the notion, most of the metabolomic variation is also captured through the first principal component of immune features. This is both an evidence that immune data is predictable using metabolomic markers and that the preprocessing steps we use for this experiment works.

Model Explanation

There are largely two tasks that machine learners tackle: classification and regression. Classification is a task to categorize a given input sample to a discrete class. Regression is the process to predict a given input to a continuous value. Since predicting an immune score is to evaluate a numeric value, the task at hand is a regression task.

In this study, we will be using a random forest to perform regression. The main rational behind using random forest is because the goal of this study is not to achieve the lowest loss on, but to identify significant biological relationship. Thus, we wanted to work with a more interpretable model, that provides a well-balanced feature importance score.

To explain how random forest-based regression works, the following section will discuss:

1. How linear regression works
2. How decision trees work
3. How decision tree-based regression works
4. How random forest-based regression works

1. Linear Regression

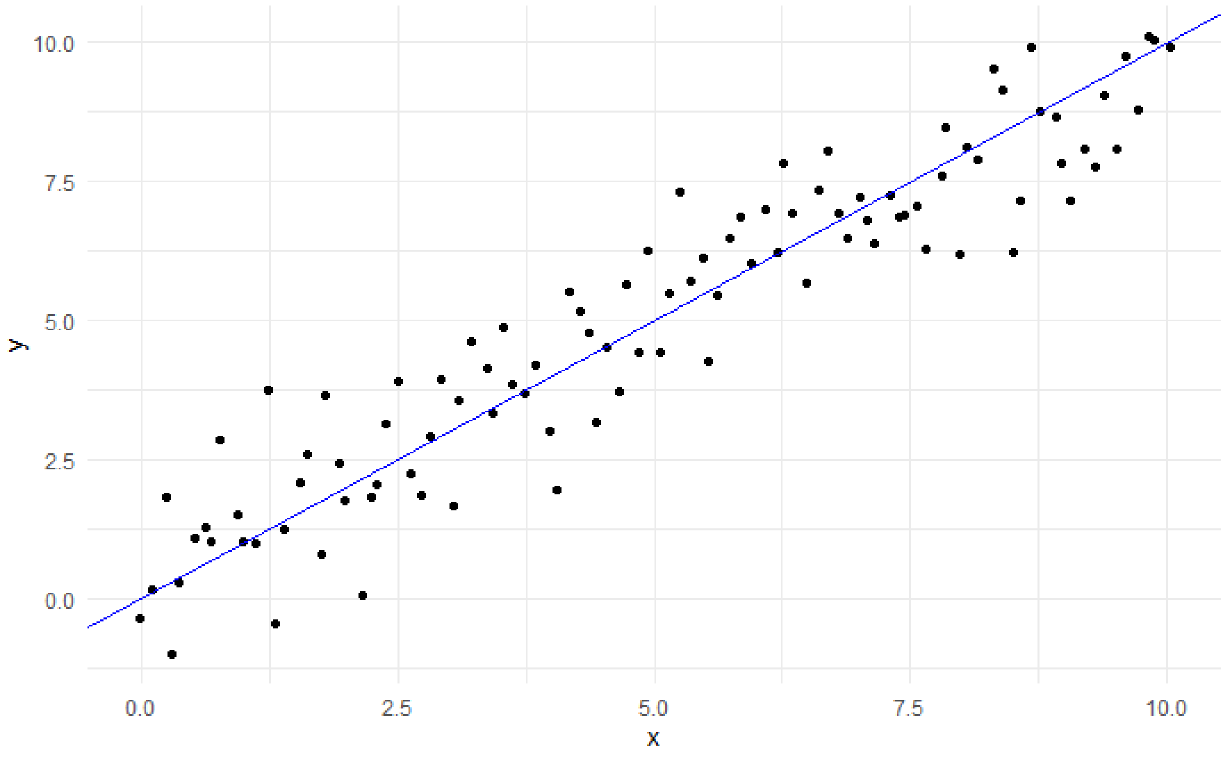

Figure 10. Plot Demonstrating Linear Regression

The simplest and yet powerful tool of machine learning on regression tasks is linear regression. The aim of linear regression is to map a set of inputs, or input vectors, into a numerical output value. As illustrated in the figure above, our goal with linear regression is to figure out the line that minimizes the distance with respect to given data points. We can achieve this task by learning the coefficients, or weights, associated with each input features.

$$\mathbf{y} = \boldsymbol{\beta}^{\mathbf{T}}\mathbf{X} + \boldsymbol{\epsilon}$$

In the equation above, $\mathbf{X}$, $\mathbf{y}$, $\boldsymbol{\beta}$, $\boldsymbol{\epsilon}$ each represent, input vector, output vector, weight, and noise term. In simpler terms $\boldsymbol{\beta}$ could be considered as the gradient. In this setup, all we have to figure out is $\boldsymbol{\beta}$ since **X** and **Y** are already known. In order to find out $\boldsymbol{\beta}$, we need to establish a way to quantify the model performance. A commonly used loss for linear regression is mean squared error.

$$\boldsymbol{L}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\mathbf{1}}{\mathbf{2}\boldsymbol{m}} \sum_{\boldsymbol{m}} (\mathbf{y} - \hat{\boldsymbol{y}})^{\mathbf{2}}$$

The next task is to find out the weights that minimize this loss. As one can expect, we take the derivative of the loss with respect to the weights to do this and solve for the weights. The ½ in above equation is an arbitrary scaling for a convenient differentiation. With a little bit of linear algebra, we can solve for the analytical solution of weights.

$$\frac{\boldsymbol{\partial}}{\boldsymbol{\partial\beta}} \boldsymbol{L}(\mathbf{y}, \boldsymbol{\beta}^{\mathbf{T}}\mathbf{X}) = \frac{\boldsymbol{\partial}}{\boldsymbol{\partial\beta}} (\mathbf{y} - \boldsymbol{\beta}^{\mathbf{T}}\mathbf{X})^{\mathbf{T}} (\mathbf{y} - \boldsymbol{\beta}^{\mathbf{T}}\mathbf{X})$$

$$= \mathbf{2X^TX}\boldsymbol{\beta} - \mathbf{2X^T}\boldsymbol{y} = \mathbf{0}$$

$$\boldsymbol{\beta} = (\mathbf{X^TX})^{-\mathbf{1}}\mathbf{X^TY}$$

This technique is good because it provides a simple and understandable solution for our input and output vectors. Firstly, this method provides with a mathematically sound and simple method for learning an approximation function. Secondly, when all the input features are normalized, the weights will tell us which features matter the most. Thirdly, since this can be calculated with a few matrix operations, linear regression is one of the computationally lightest algorithms as well. The strengths of this model lie in simplicity and interpretability.

However, there are a few downfalls of this method. Firstly, as indicated in the algorithm's name, linear regression is incapable of learning non-linear relationship. Secondly, the algorithm is prone to outliers. An outlier within the dataset can mess up regression when using mean squared error metric, since one large square can outweigh many small squared values when averaged. The last disadvantage of linear regression is that the algorithm is not scalable to number of features. As the number of dimensions increase for a linear regression, the algorithm becomes much more susceptible to overfitting.

2. Least Absolute Shrinkage and Selection Operator (LASSO) Regression

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j| = \mathrm{RSS}+\lambda\sum_{j=1}^{p}|\beta_j|.$$

Equation. LASSO Regression

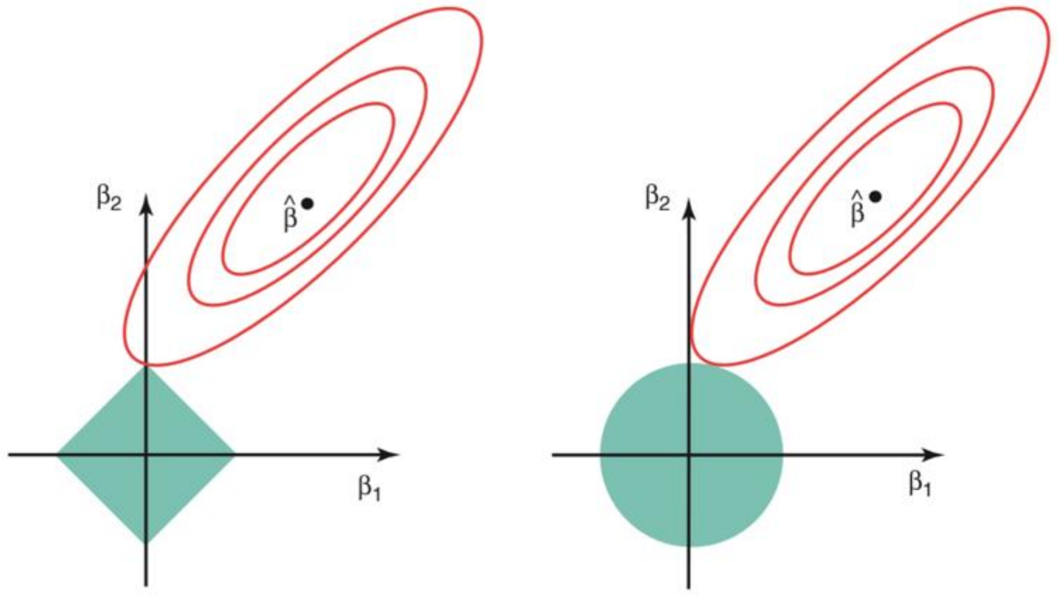

Figure 11. Figure Demonstrating Loss Contour of LASSO Regression[2]

LASSO regression is at the bottom heart linear regression with an extra term to penalize weights that matter less to the prediction. The bottom line for LASSO regression is bias-variance tradeoff stating that allowing small bias decreases variance of the prediction by a factor of square. LASSO regression, also known as L1 regularization, penalizes the all the weights in the L1 space as shown in the left of Figure 11. The biggest strength of this model is that it can remove a lot of the features that do not matter to the prediction. As in

[2] Elements of Statistical Learning Figure 3.11

the figure, one can observe that the turquoise square meets with the loss contour at an edge. This indicates that the optimal LASSO coefficient sets b_2 as 0. On the right is ridge regression, also known as L2 regularization. While ridge regression is great at penalizing very large weights,

3. Decision Tree

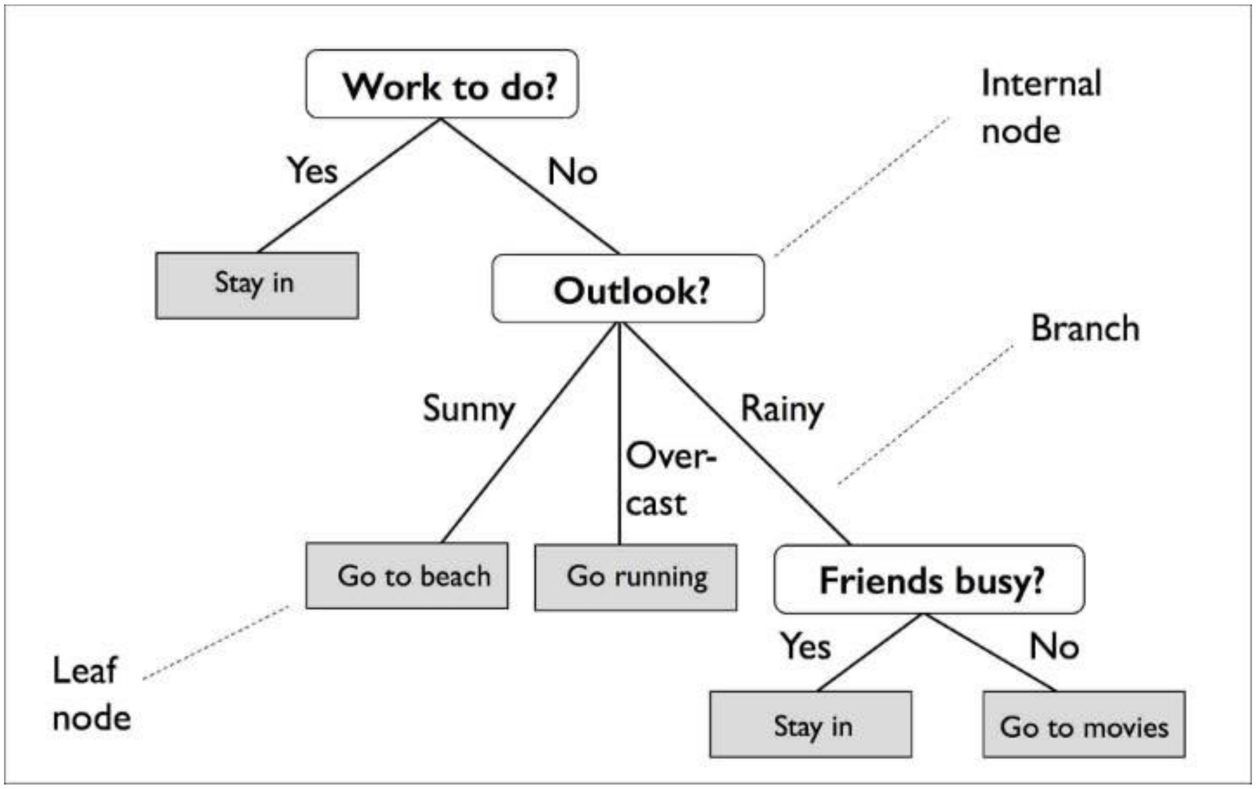

Figure 12. Demonstration of a Decision Tree

Decision trees are machine learning models that partitions data into several bins or clusters called the leaf node. As its name implies, the model builds a tree to split data points into several regions according to a series of criteria. There are three common ways split criteria is decided.

Three common ways to decide split criteria for decision tree:

1. Gini Impurity
2. Information gain
3. Variance Reduction

Gini Impurity and entropy are ways to come up with decision trees for classification, whereas variance reduction is a metric for regression trees. Entropy, at an information theory perspective, is a quantification of degree of chaos within a system. Similarly, information gain is described as how much degree of chaos we remove from the system. The mathematical formulation of information entropy lies under log probability domain and is defined as $H(X) = -\sum_i p\log(p)$. To give a bit more context, let us examine the example of a deck of cards. The information about the card "Ace of Spades" consists of the information "Ace" and "Spades." While these two pieces of information add up together, the probabilities associated with this information multiply to form "Ace of Spades" in the probability space (1/4 x 1/13 = 1/52). Hence, one conveys more information when he or she is more specific when describing a system. For instance, information content from description "Spades", "Ace", to "Ace of Spades" are in increasing sequence as specificity increases or probability decreases.

The same metrics could be used when deciding which split criteria to use on which branches in the case of decision trees. Decision trees come up with several split criteria and decides which yields maximum information gain. If a decision tree were to bin a deck of card into different bins, the first split should be the value (numbers and letters), and the second split should be the suites. In this case, each of the cards will fall under a leaf node and the tree can be considered a overfitted tree because all the samples are a leaf node.

Gini impurity is much similar metric to entropy. The mathematical formula for Gini impurity is $Gini\ Impurity(X) = -\sum_i p(1-\mathrm{p})$. This is a more commonly used metric for decision tree classification for computational reason. Gini impurity is the first order Taylor Series approximation for entropy. As the formula comes in a polynomial form, Gini impurity is a much more computationally effective metric for calculating a derivative. For the case of regression, the metric used is the overall variance. This makes sense as the data is split into several clusters. The most important advantage of decision tree is that the model is highly interpretable. As shown in the Figure #, one can look at the dendrogram and immediately understand what is going on within the model.

4. Tree Based Regression

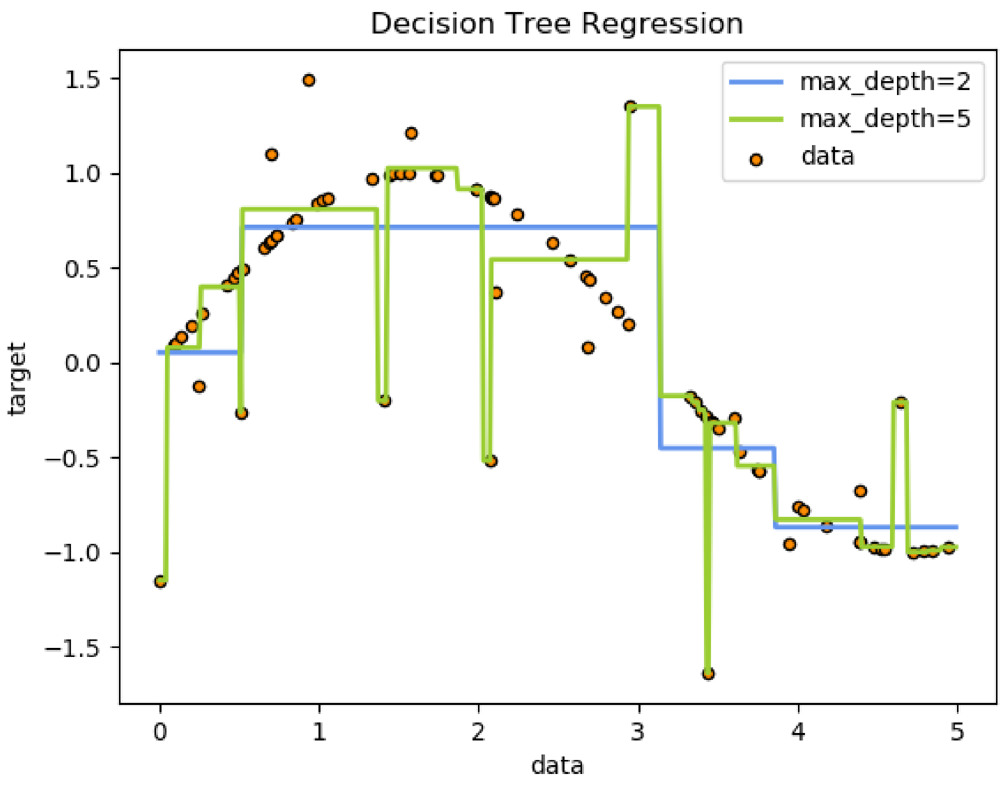

Figure 13. Demonstration of a decision tree-based regression

As mentioned in the previous section, one of the downsides of decision trees are that they are prone to overfitting. One of the ways to combat this is through limiting the depth that the tree can grow. Decision tree regressions are done either through a linear regression or a simple mean operation at the leaf nodes. Figure # shows an example of decision tree regressor with two different max_depth hyperparameter. As one can observe in the plot, the fit with max_depth of 2 severely underfits the data while the other severely overfits the data. Due to this sensitivity of hyperparameter that is hard to control, regressions are usually not done through decision trees.

5. Random forests

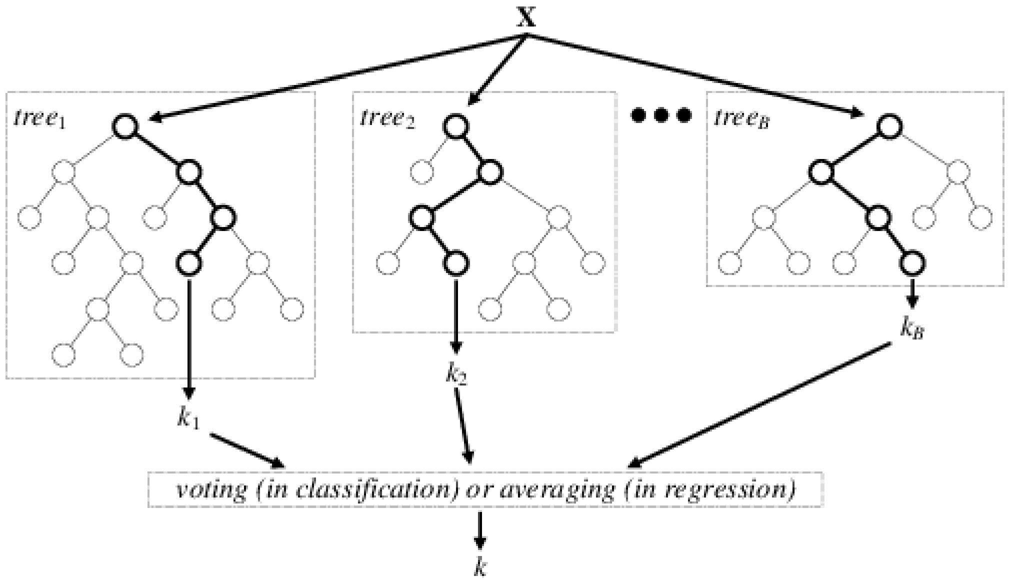

Figure 14. Random Forest based Regression

The main idea behind using random forest is that while a single decision tree could be overfitting and hyperparameter dependent, multiple trees can overcome these flaws much better. The technical term for the methodology is called bootstrap aggregating. The aggregating means that we are either taking the max vote for classification or averaging for regression as our final predictions. Bootstrapping selects different permutations of feature subset to build a tree so that every tree that we build are different. This model tends to be much more balanced and its predictions tend to be much smoother compared to the flat lines of the decision trees for a given region. For such reason, it is one of the models that are particularly good at good at feature selection.

6. Spearman Correlation test for multiple hypothesis testing

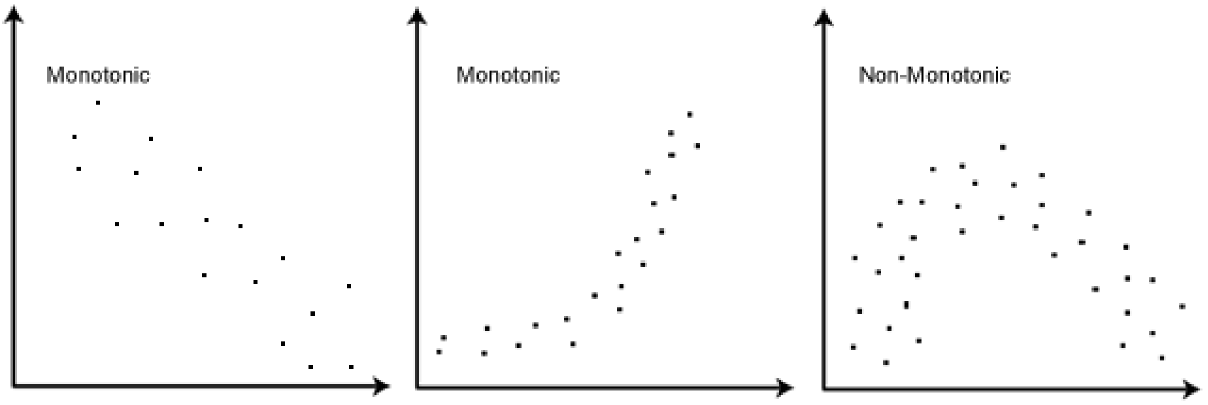

Figure 15. Monotonic vs. Non-monotonic Relationship to explain Spearman Correlation

The most known form of correlation is Pearson correlation, which is defined as $\rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y}$. Whereas Pearson correlation is useful at capturing correlation between two variables for the relationship on the left of Figure 15, it is not the best tool to capture the relationship in the second plot. In other words, the downside of Pearson correlation that it is not at capturing monotonic relationship like exponential or log distributed correlation between two random variables. To overcome this issue, we can use spearman correlation to measure correlation through ranks rather than in the original variable space. The equation for this is $\rho_{\mathrm{Spearman}} = 1 - \frac{6 \sum d_i}{n(n^2-1)}$ where d_i is the difference between the ranks for two variables. We use Spearman correlation with this experiment because the metabolomics file that we received were already quantile transformed during its preprocessing stage, and thus the scale might be arbitrarily transformed.

Results

1. Using Metabolomics data to predict Immune features

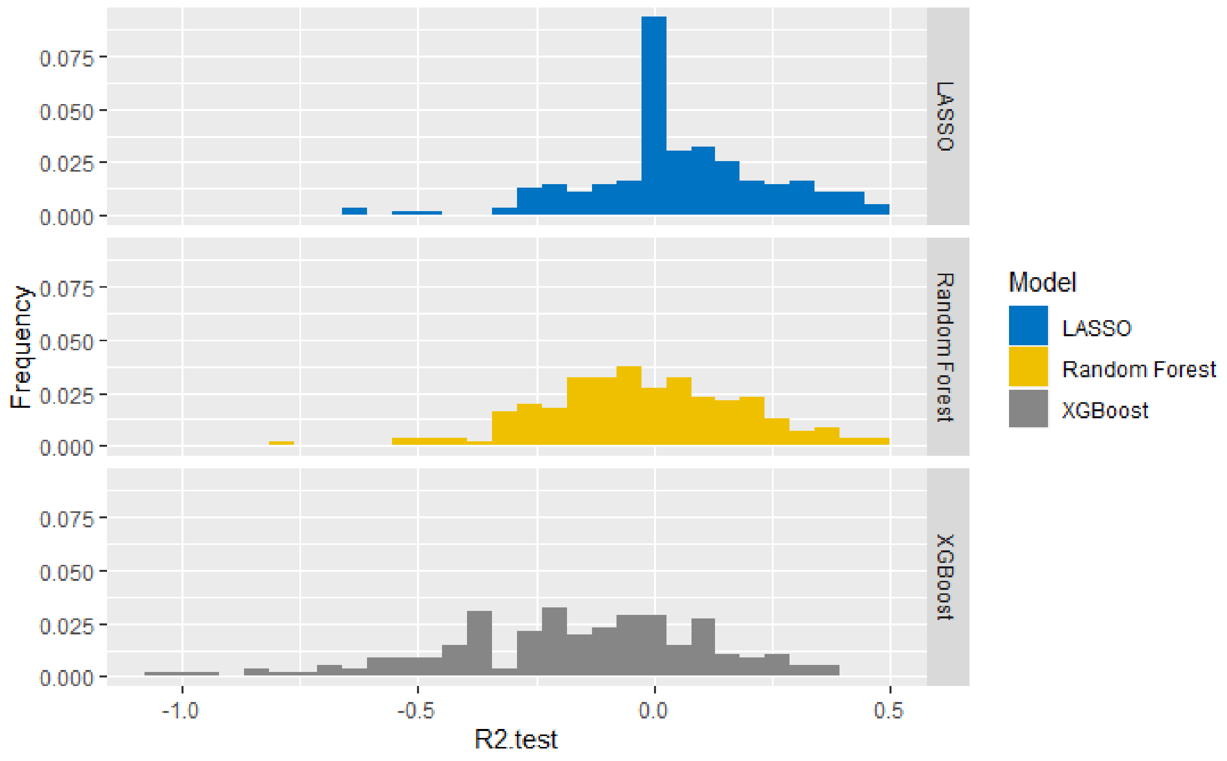

Figure 16. Histogram of Performance (R^2) for each Machine Learning Models

As mentioned previously, we wanted a model that was interpretable. For such reason, we selected three models in advance: LASSO, random forest, and XGBoost. LASSO coefficients provide linear relationship between each of the input features with the output. Random forest and XGBoost libraries commonly provide a feature importance score based on Gini impurity in the classification task and variance reduced in the regression task. While the other two models cannot tell apart whether relationship between a feature and output is correlated or anti-correlated, they still provide some insight into which features matter more than others.

The first point of comparison was the performance of each models. The figure above plots the histogram of test R^2 scores for each of the target immune features. For interpretability purposes, R^2 greater than 0 means the prediction is better than the plain mean. Thus, one can interpret results that are less than 0 as unpredictable and greater than 0.2 as predictable. As demonstrated in Figure #, the model that performs best is LASSO. There are a few reasons why LASSO performs so well on this dataset. First is that this dataset is in a domain where number of samples are approximately similar to the number of features. LASSO regression works well in this domain because it removes most features that do not matter to the prediction. Second is the fact that these relationships are mostly linear. From this point on, we decided to use LASSO for both tasks as LASSO regression was the model that performed the best on both tasks and also provided the additional sign that indicates whether an input feature is a promoter or an inhibitor of the target feature.

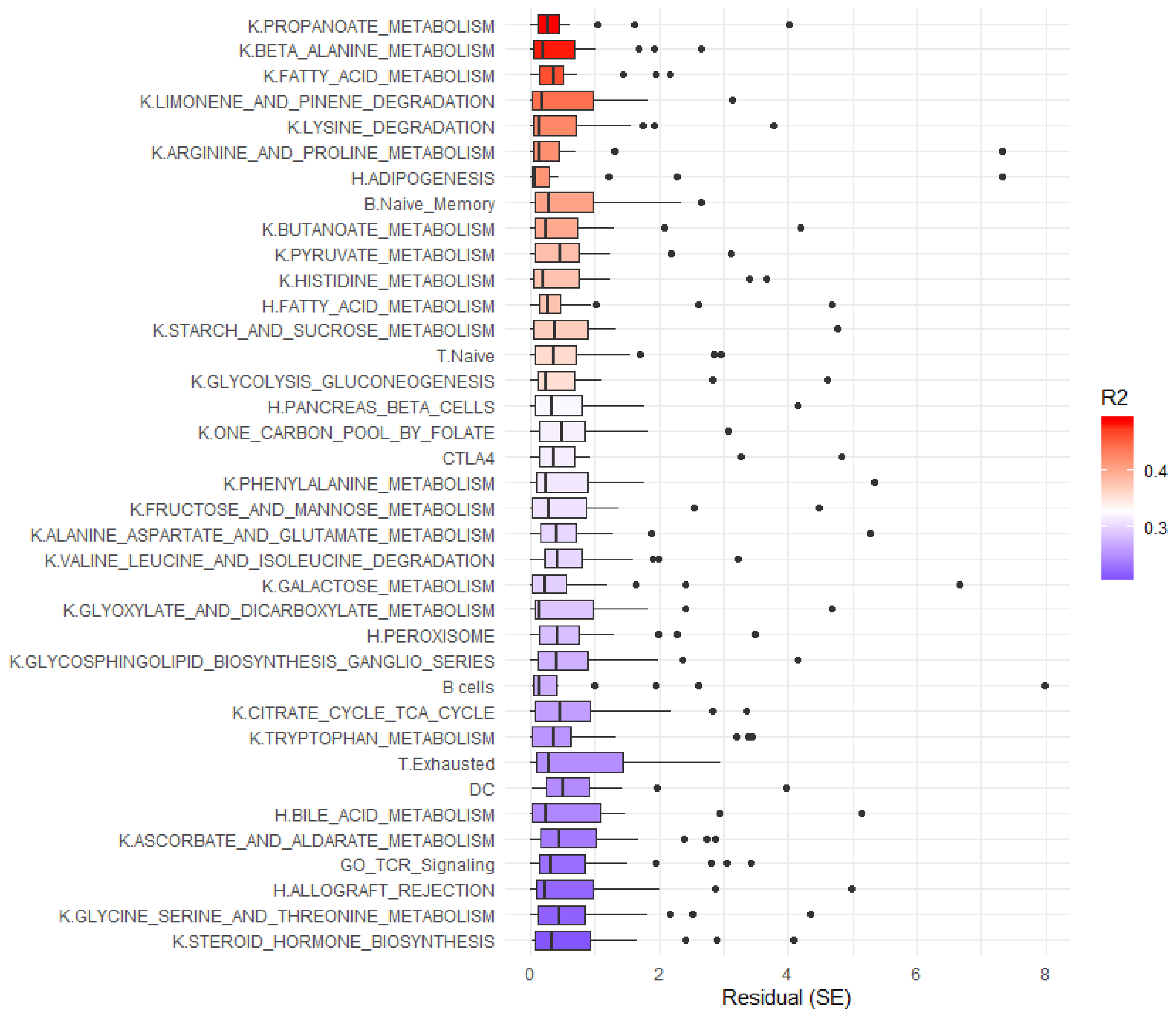

Figure 17. Test set residuals from the LASSO Regression

The figure plots the resulting squared errors for each of the prediction metabolites make on immune features. The results were filtered for immune features that had greater than R^2 of 0.2. The results were sorted in decreasing predictability order. As it can be seen in the plot, major factor that sets apart the good predictions from others is the outliers. Our top results are prediction of propanoate metabolism and beta alanine metabolism etc.

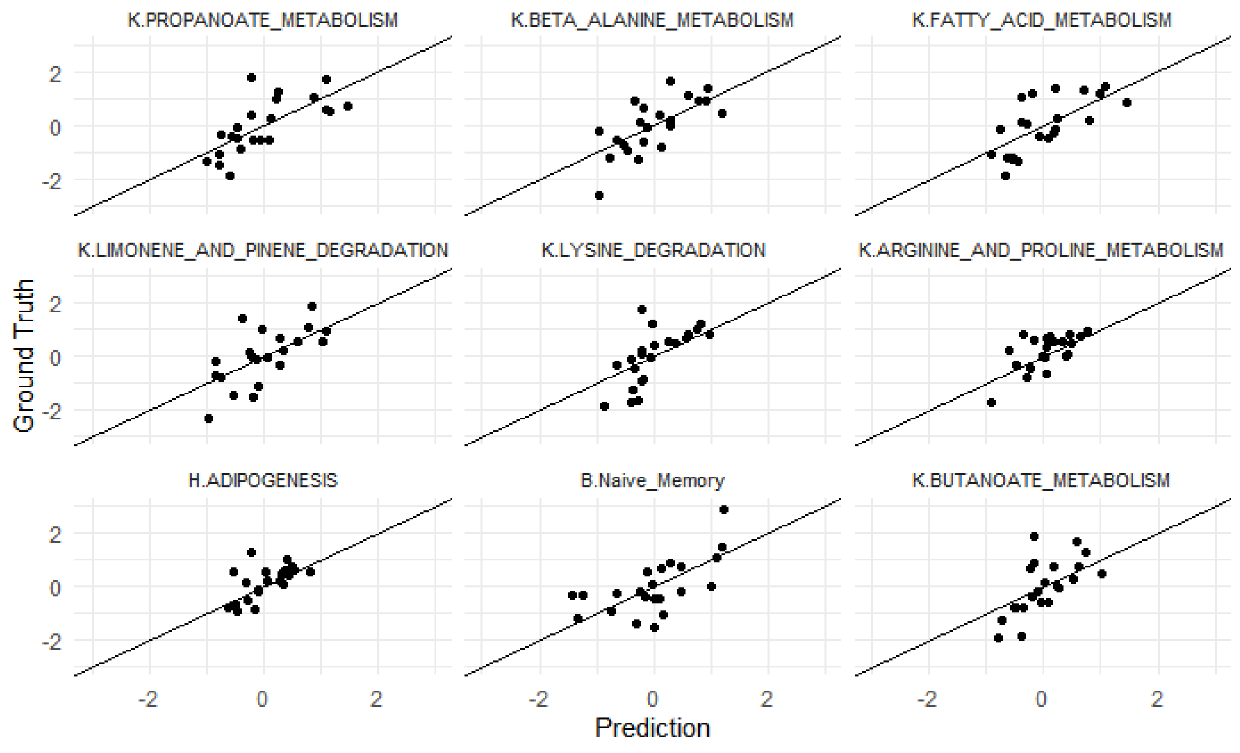

Figure 18. Comparison of Prediction and Ground Truth for Best Prediction Results

These are some of the best prediction results from the immune scores, specifically the 9 immune features with the lowest mean squared error. As one can see in the diagrams, all the truth-prediction pairs are close to the line with slope 1, which is where all the data points should lie on.

LASSO Coefficients and Correlation Coefficient

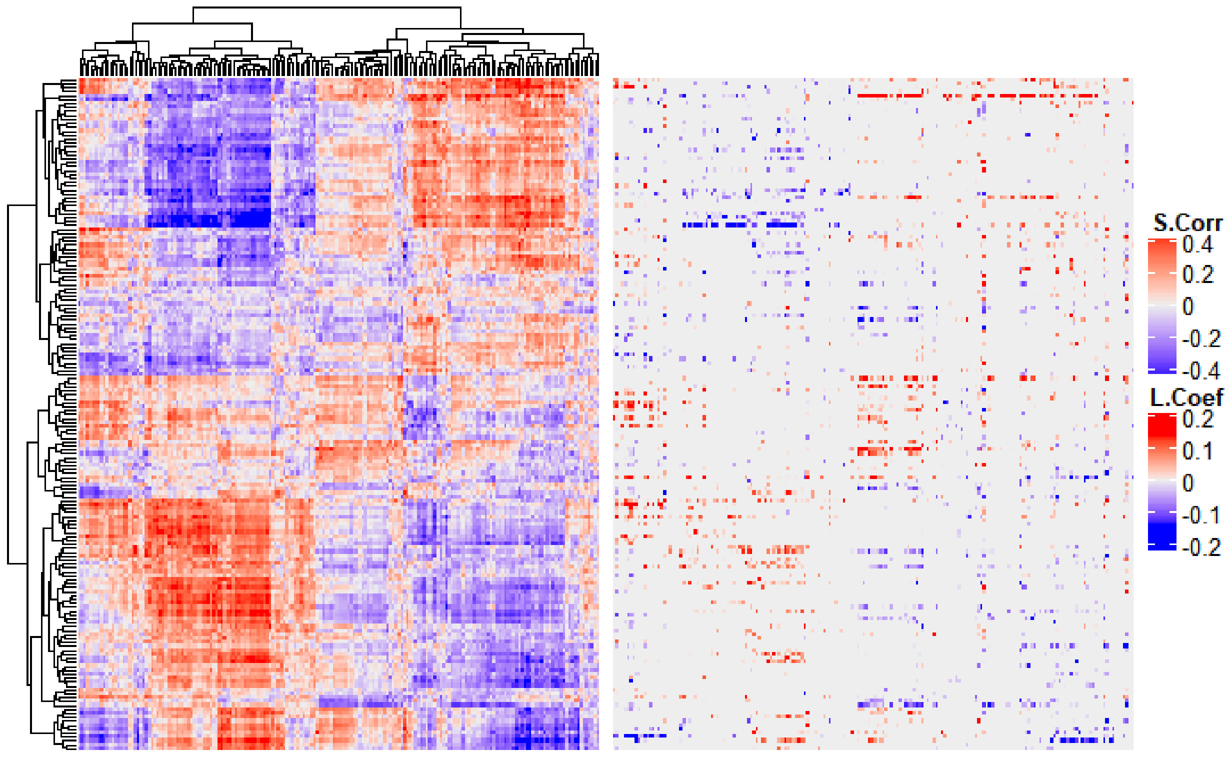

Figure 19. Pairwise Spearman Correlation (left) and Lasso Coefficient (right) between metabolites and immune features

The figure above plots correlation coefficient on the left side, and respective LASSO coefficients on the right. Each of the rows indicate metabolites and the columns indicate immune markers. The ordering of the rows and columns were decided by a hierarchical clustering regime through Spearman Correlation, and the same exact order was used for LASSO coefficient. As juxtaposed from left to right, we see that the signs or the colors of all regions match as expected. One overarching difference between the heatmap on the left and right is that the LASSO coefficient appears to be much sparser compared to the spearman correlation. This demonstrates one of the reasons why LASSO coefficients are much more valuable than spearman correlation: it achieves feature selection through L1 regularization that removes most irrelevant features and leaves just enough to reconstruct the original data. As most of the rows and columns are hardly interpretable with this many feature, we decided to distillate a few of the rows and columns that show much more dense information about which columns and rows are important. Yet, one important takeaway from this plot is that the row structure is more conspicuous than column structures for LASSO coefficients. This indicates that a few metabolites are important for multiple immune features.

Important Metabolic Markers

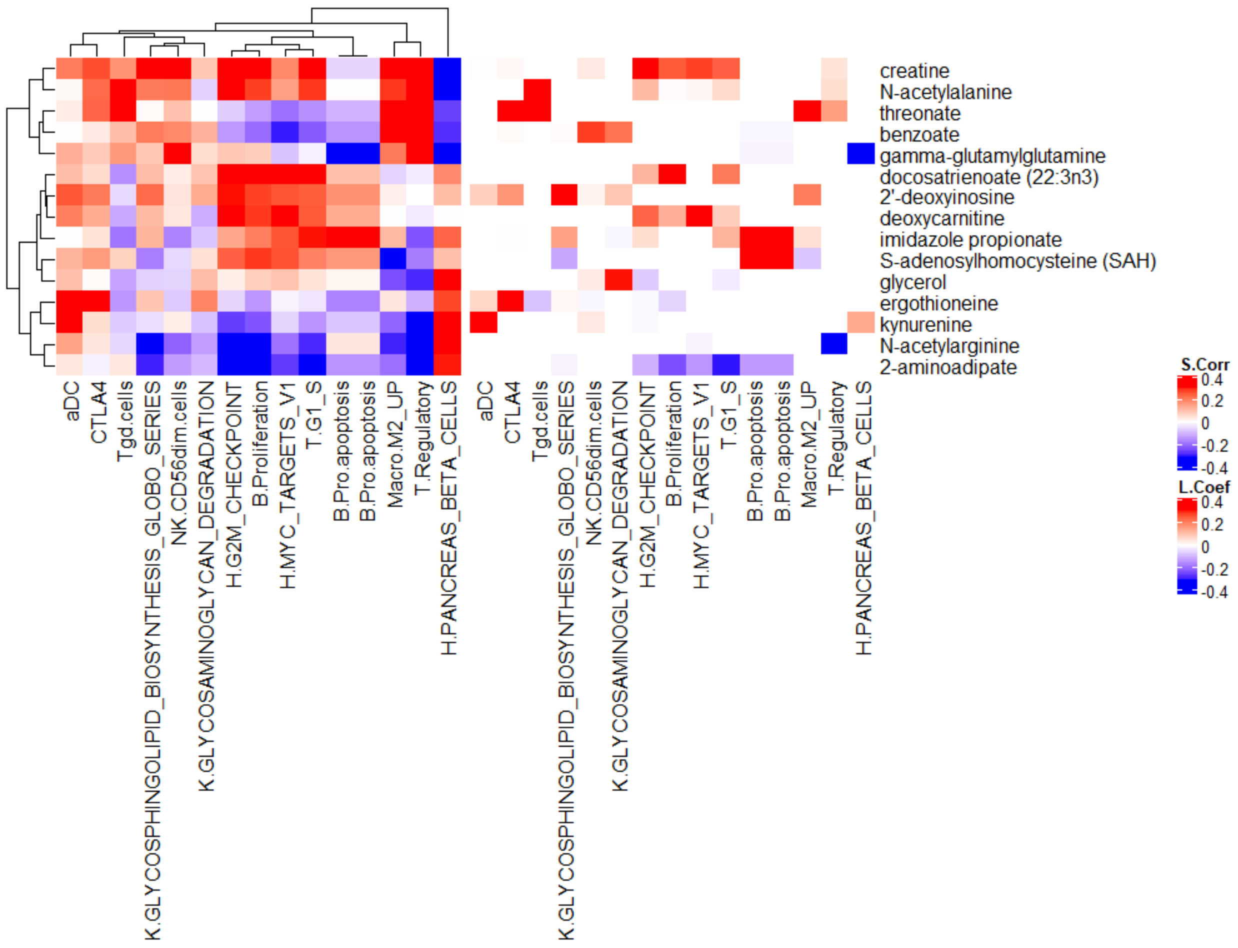

Figure 20. Highest LASSO Coefficients and their respective Immune Features

We specifically use two algorithms to come up with a condensed correlation and lasso coefficient matrix: first is to fetch metabolites with top-K LASSO coefficients, and get which immune feature it predicts well; second is to fetch the top-K performing immune features and retrieving their metabolites with highest LASSO coefficient. One point to note in this figure is that the highest LASSO coefficient is often not the most important thing to consider for two reasons. The magnitude of a single coefficient with respect to others matter given a single target, and the target R^2 score also matters as some of the fetched output might not be the most predictable feature.

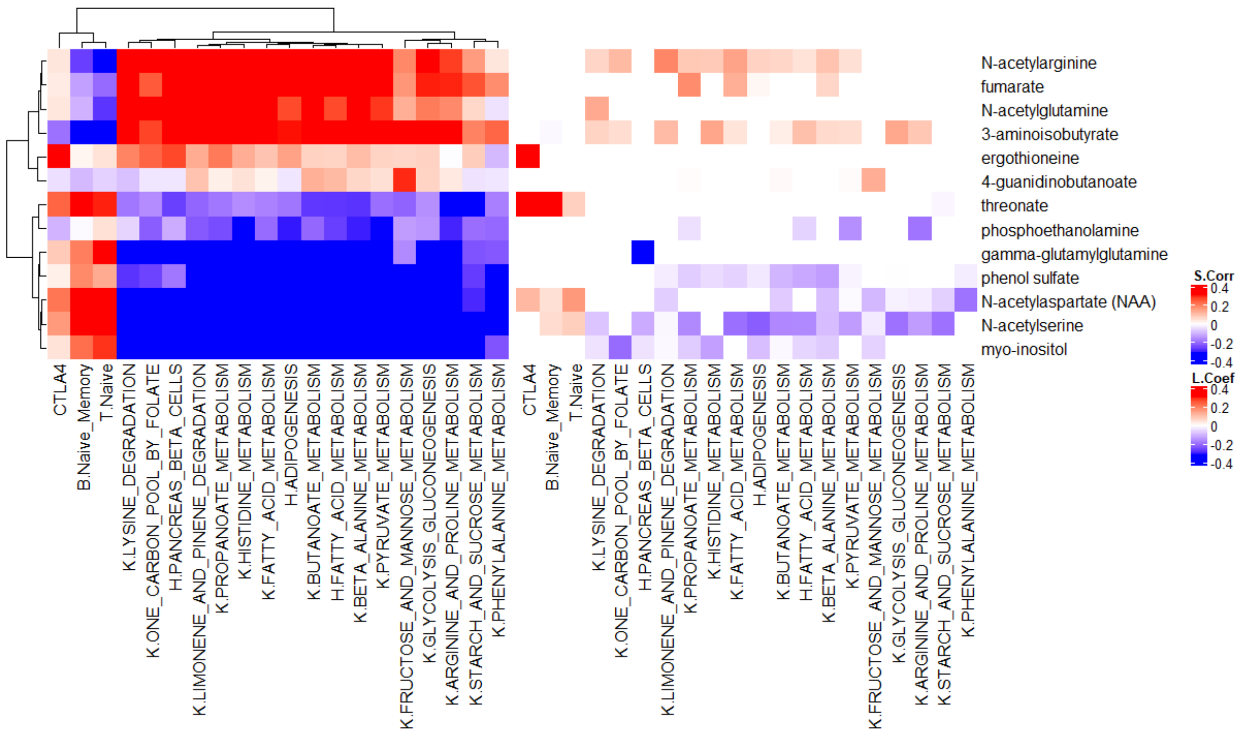

Figure 21. Most Predictable Immune Features and their highest LASSO Coefficients

Continuing from the previous result, this plot shows most predictable immune features and their metabolites with highest LASSO correlation. One of the notable results is that the most predictable immune features are mostly collinear. As in Figure #, the first principal component explains 30.56% of the variance. This indicates that the features that are most predictable are mostly similar immune features. Compared to the previous plot, the LASSO coefficient in this plot has much lower values and are much more spread apart many different metabolites. One notable thing is that a lot of the metabolites like N-acetylarginine, threontate, and ergothiononine occur in both plots.

2. Using Transcriptomics data to predict Metabolomics Data

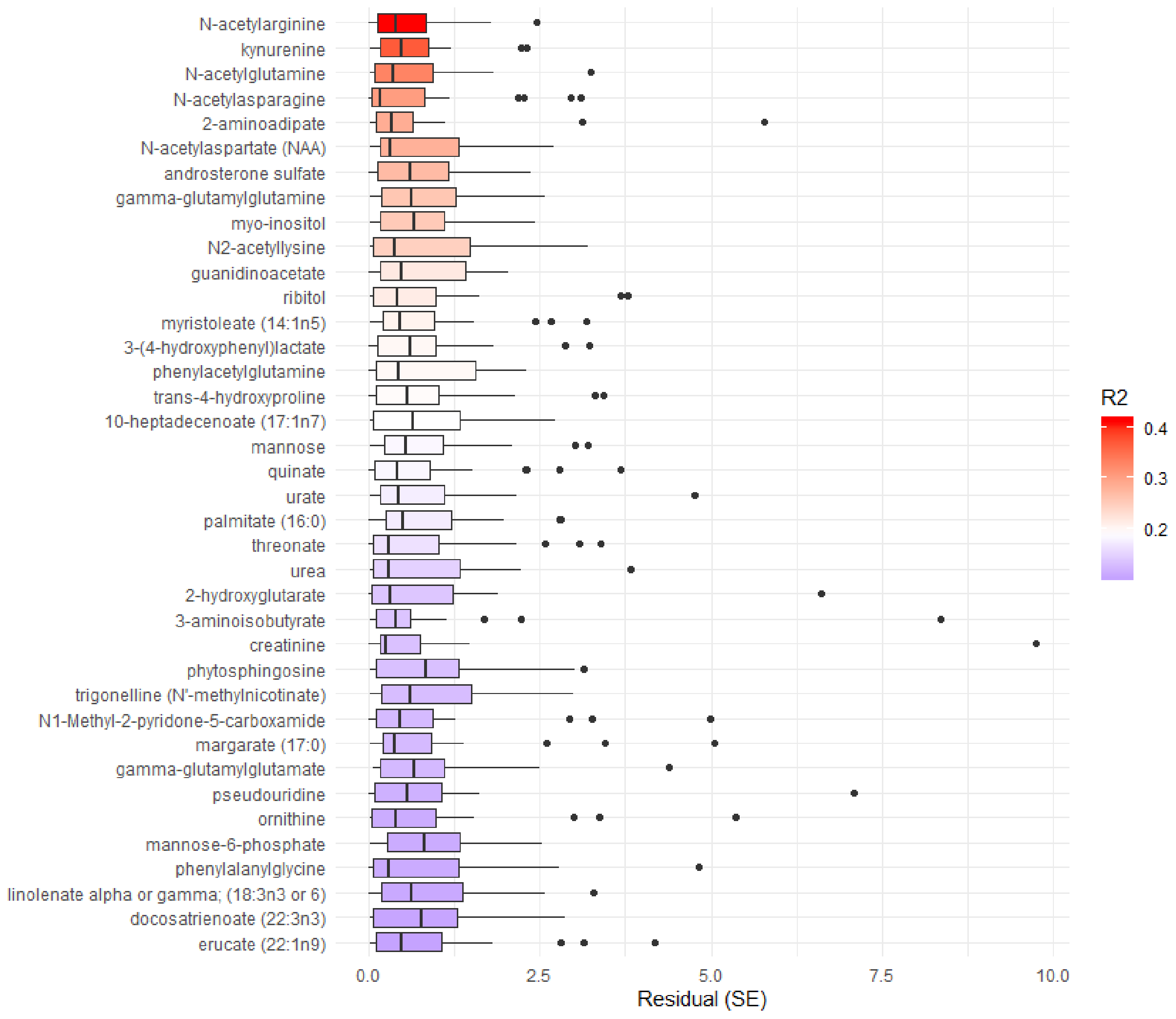

Figure 22. Test set residuals from the LASSO Regression

The same analysis for the first part was also done on the second part. The top predictable metabolites were the N-acetyl groups, kynurenine, 2-aminoadipate, myo-inositol etc. About 10% of the metabolites had greater than 0.2 R^2 score, and we considered them predictable. For all the predictions, most fall under reasonable bound except for a few outliers. As we show in later figures, some of these metabolites are predictable by a very few metabolites. One of the evidences that validates the prediction works is that the highly correlated metabolites are what makes the genes.

LASSO vs Correlation Coefficient

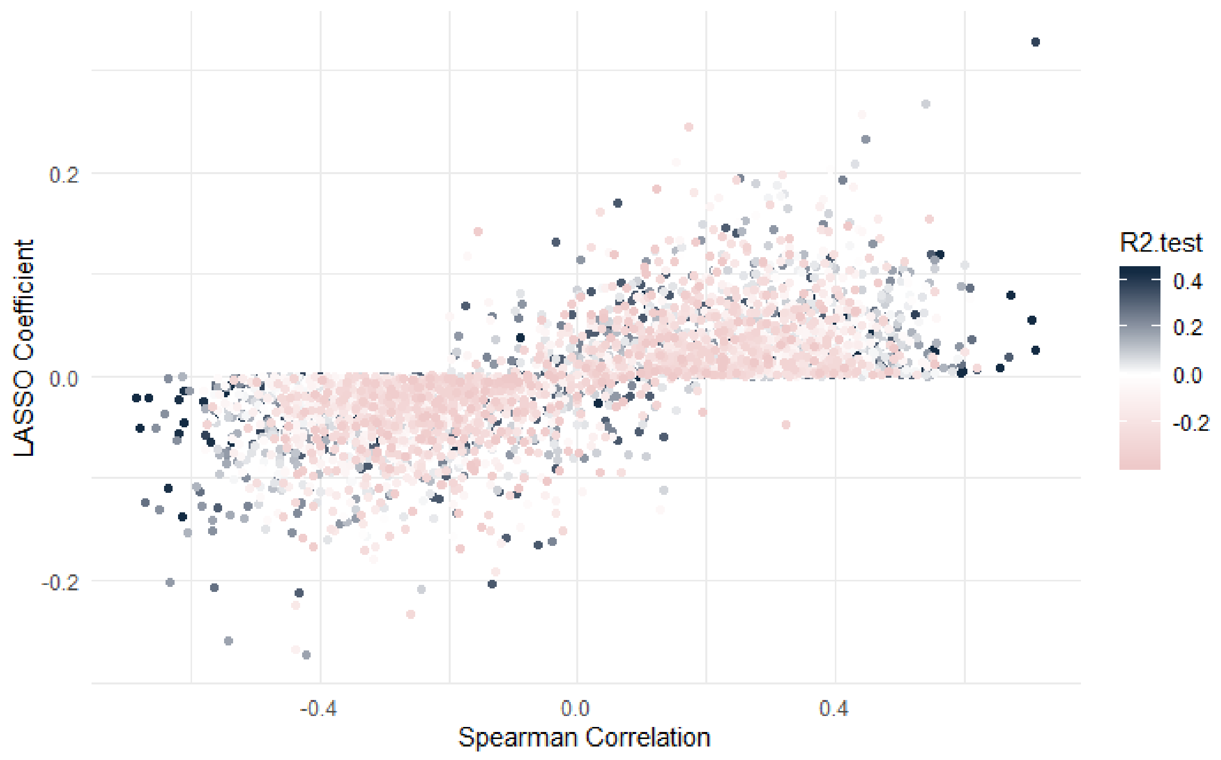

Figure 23. Spearman Correlation vs. LASSO Coefficient

X-axis of the plot above shows the Spearman Correlation for each gene-metabolite pair and the y-axis shows the respective LASSO coefficients. As in the plot, these values are correlated. The LASSO coefficient also tends to pick up much less features compared to the spearman correlation and thus is in a much more condense form. While the features that are lower than 0.5 correlation are mostly unpredictable, the ones that have greater than 0.5 correlation are mostly predictable.

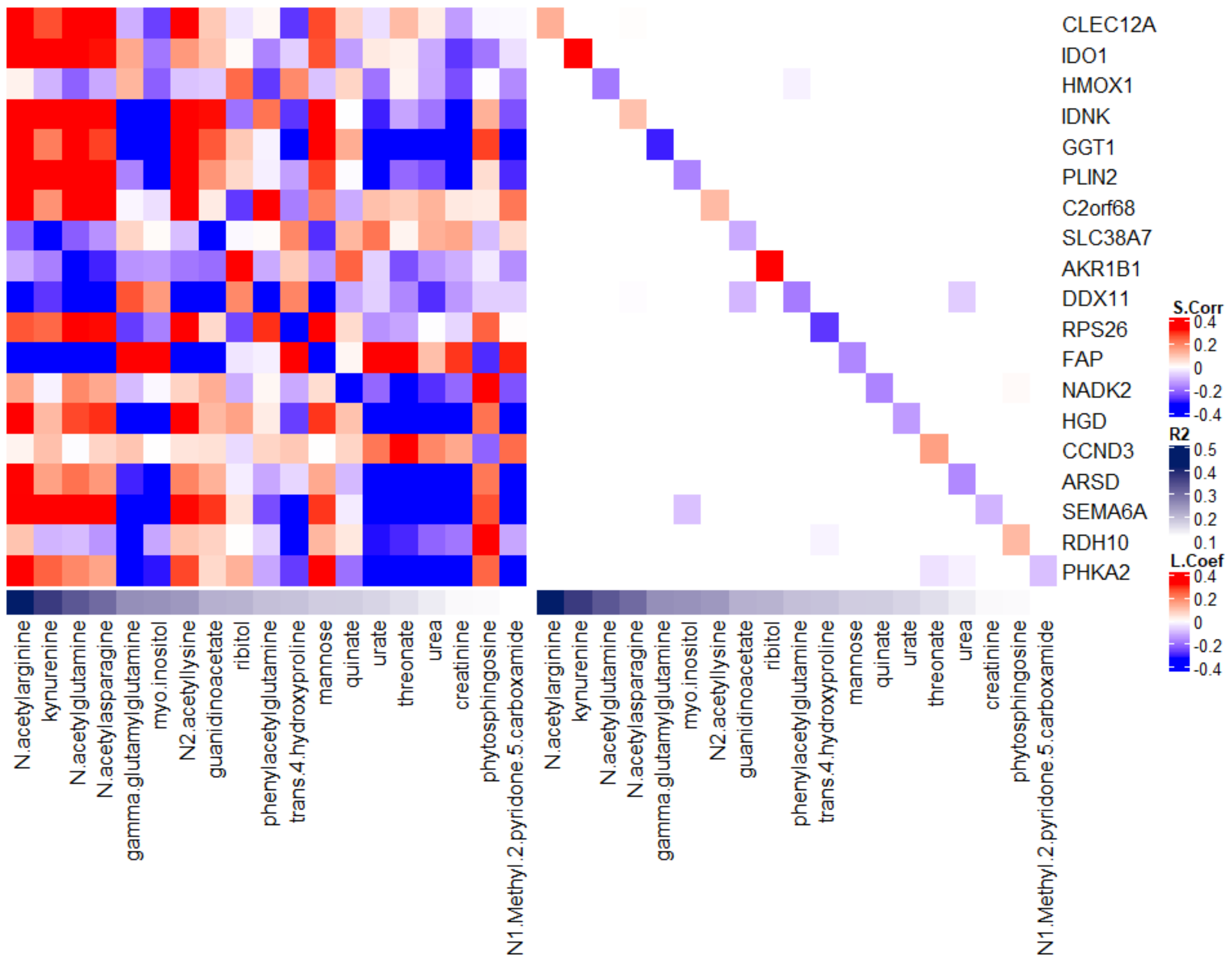

Figure 24. Most Predictable Metabolites and their highest LASSO Coefficients

As with part 1, we report the most predictable metabolites and their most predictive genes. We skipped the highest LASSO coefficient plot, because the information in the plot were, while also confirming, somewhat redundant. There are three promising relationships that the LASSO coefficients pick up. First is the relationship between kynurenine and IDO1, second between gamma glutamylglutamine and GGT1, and third between ribitol and AKR1B1.

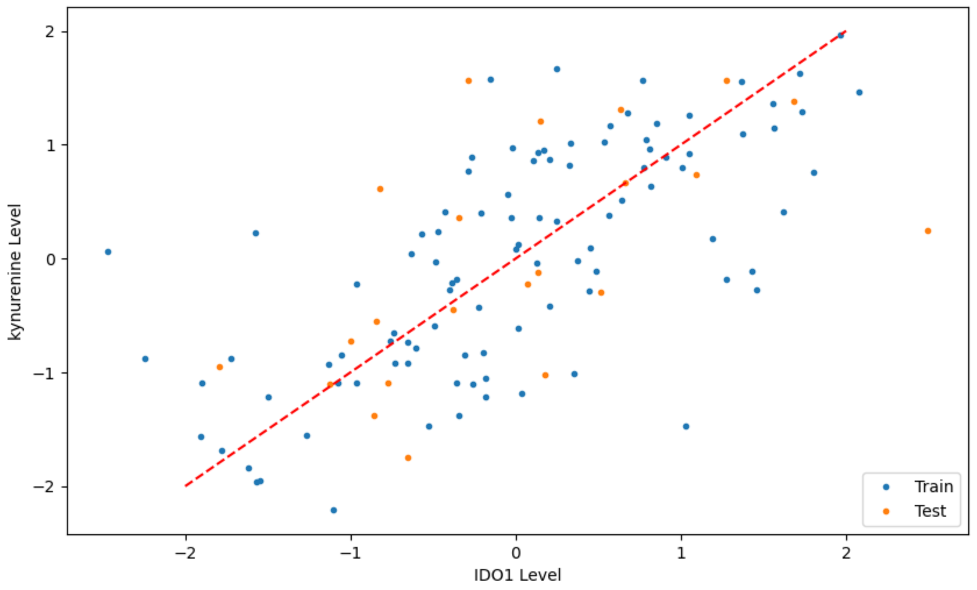

Figure 25. IDO1 level vs. Kynurenine Level

IDO1 catabolizes tryptophan to produce kynurenine to exert immunosuppressive functions suppressing effector T and natural killer cells and promoting neovascularization of solid tumors. This behavior is happened to know in RCC as well and is one of the well-practiced targets of immunotherapy. Similarly, GGT1 is a gene that triggers catabolism of gamma glutamylglutamine in chromophobes. AKR1B1 is also a gene that is known to produce AKR1B1 in clear cell papillary. Although the latter two has not been specifically studied within the RCC environment, it is highly likely that these are true positives.

Given the observations above, one of the striking and promising result from the figure above is the high predictability of N-acetyl groups. To this date, there has been no study or report mentioning the importance of N-acetyl groups in mammalian metabolism. The fact that we pick up these signals may be a biologically important discovery.

3. Pancancer Analysis

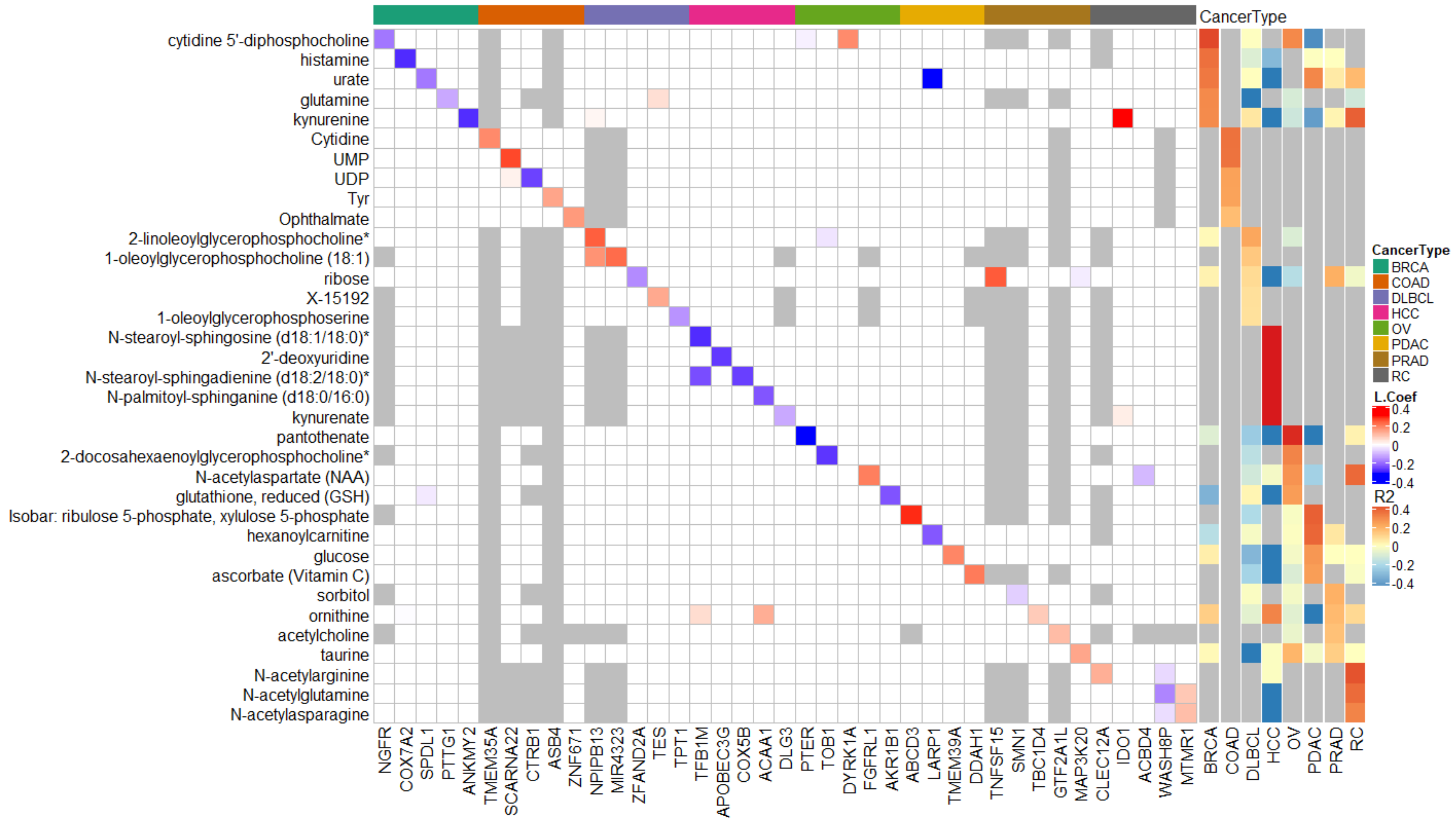

Figure 26. Pancancer Prediction Results and their most significant LASSO coefficients

Conclusions

In this experiment, we explored various machine models to illustrate the concept of in silico metabolomics: that the level of metabolites could be imputed based on readily available transcriptomics data for cancer samples. From the prediction results, we were able to identify several genes that were highly indicative of metabolites in cancer microenvironments. Most notable ones, especially given that our kidney cancer data is the richest dataset, were between IDO1 and kynurenine, AKR1B1 and ribitol, and GGT1 and gamma glutamylglutamine. We extended this analysis to pancancer analysis, across 8 different cancers. Of these, one of the most staggering result is between AKR1B1 and glutathione, reduced (GSH) in ovarian cancer. This is a similar metabolomic relationship between glucose-transforming polyol pathway and gene AKR1B1.

One of the reasons why LASSO regression was successful was because the data was in high-dimension low observation regime and the relationships were linear. Nevertheless, there were some challenges in this experiment. The two most challenging part of the experiment was the lack of quality and the lack of observation, literally, of the data. Both metabolomics data and transcriptomics data undergo multiple stages of preprocessing, which each layer adds and further propagates error. The metabolomics data are known to have batch effect coming out of chromatography machine. It goes through a preprocessing pipeline where missing values are imputed, and quantile transformation is applied. These all result in a lower the signal to noise ratio at the end of the day. The low signal to noise ratio could potentially be counterbalanced by more observation. We have trained our ccRCC model with 99 samples and tested it on 22 samples, from three different cohorts.

While this is phenomenally abundant for statistics study, this is barely on the edge of possible for a machine learning project this complicated. It would have helped if we had more studies and samples.

We look forward to further studies that could investigate not only the relationships in what predicts better metabolites but also using known pathways to investigate the chain reactions going on in the microenvironment in the computational field, and further validation of these hypothesis in vitro and in vivo so that the discovered relationships could be further used in clinical trials and practices.

Appendix

Appendix A. List of Results for Pancancer Analysis

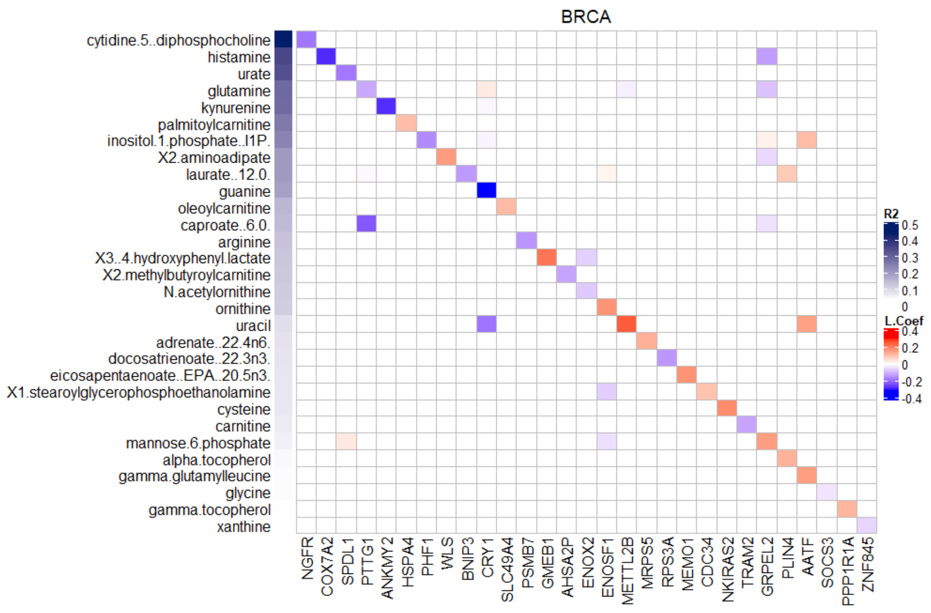

Figure 27. BRCA LASSO Coefficients

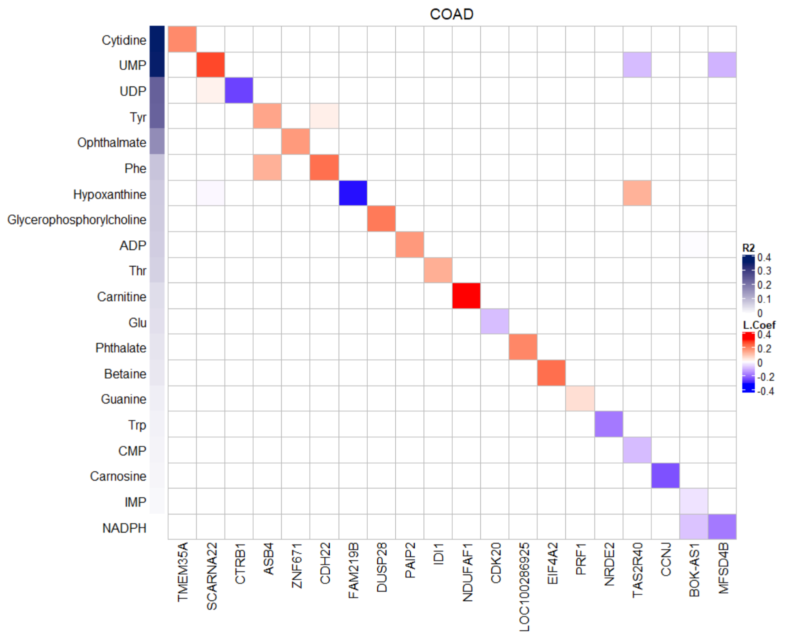

Figure 28. COAD LASSO Coefficients

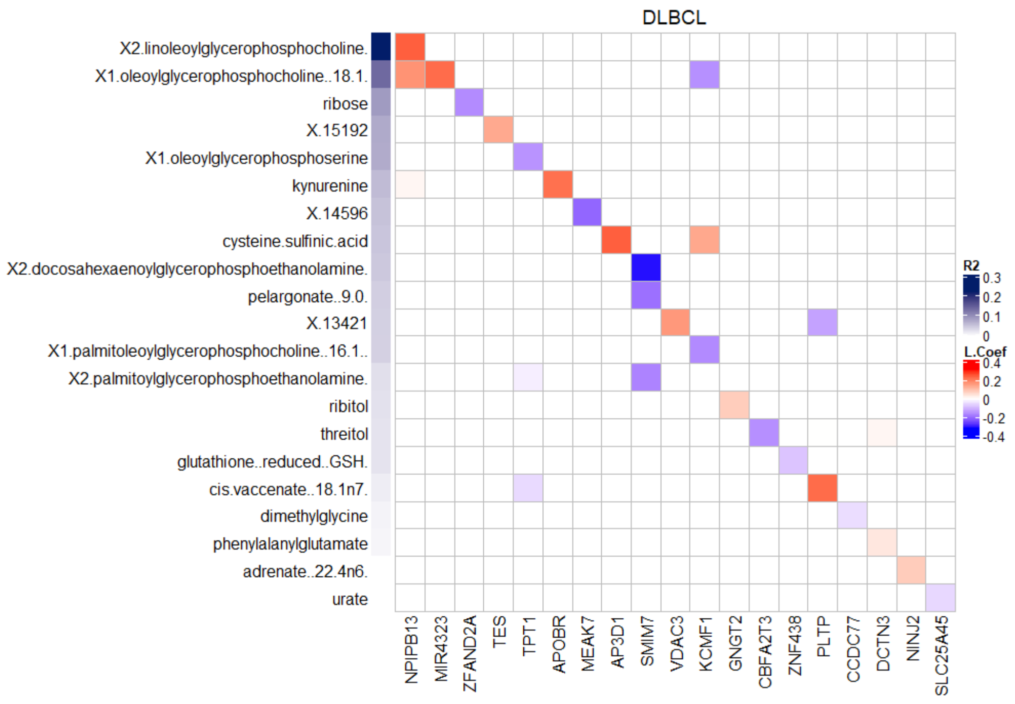

Figure 29. DLBCL LASSO Coefficients

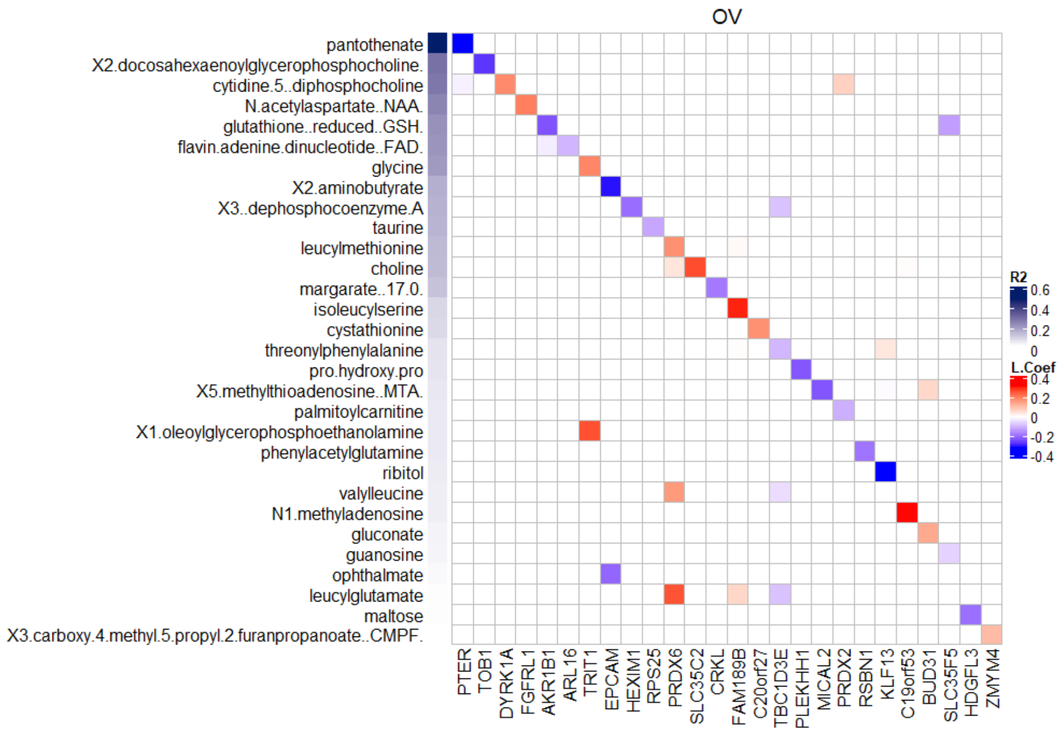

Figure 30. OV LASSO Coefficients

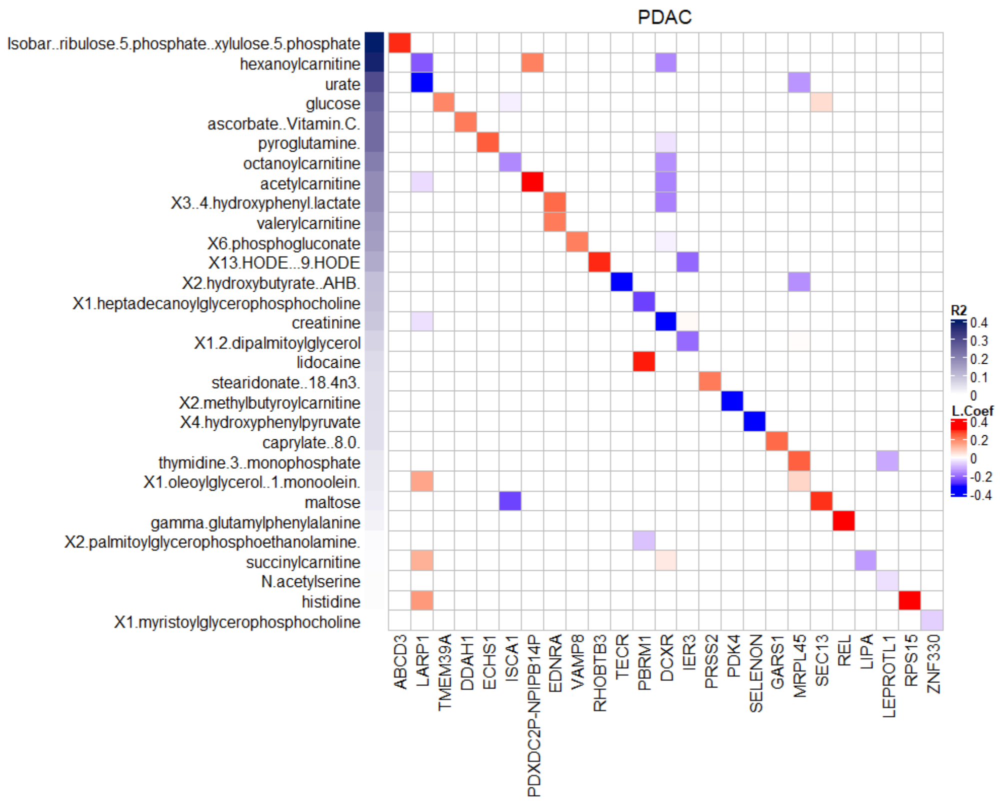

Figure 31. PDAC LASSO Coefficients

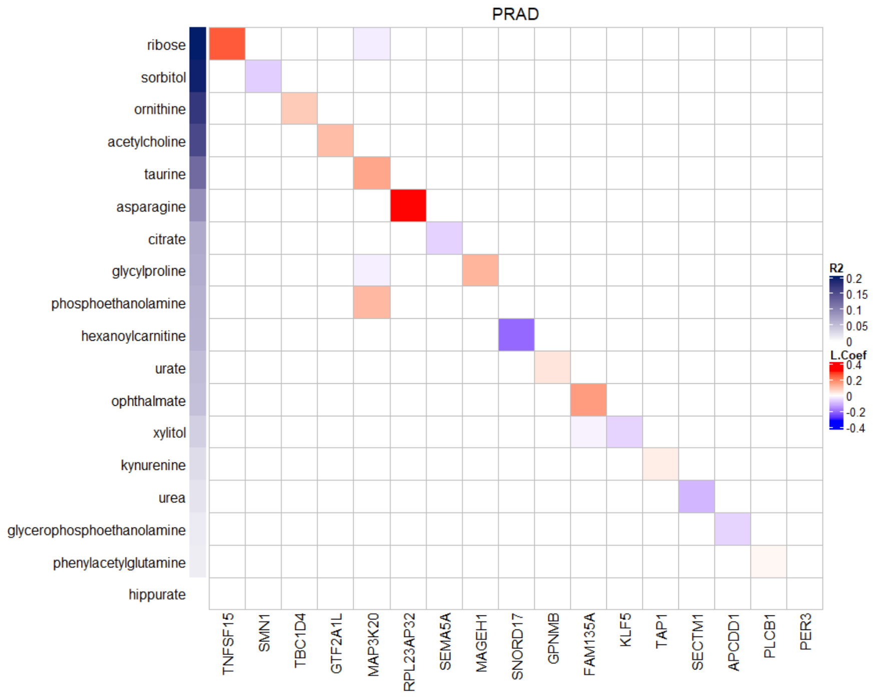

Figure 32. PRAD LASSO Coefficients

Appendix B. Code

Loader.py

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, RobustScaler, PowerTransformer
from sklearn.model_selection import train_test_split


class Base:
    def __init__(self, filename = None, lowpass = True, threshold = 0.1, sc = "z-score", with_mean = True, with_std = True, log_transform = False, mean_filter = 2):
        if filename:
            # read file
            self.data = pd.read_csv(filename)
            self.scaler = None

            # apply lowpass filter
            if lowpass:
                passed = self.data.apply(np.var) > threshold
                self.data = self.data[self.data.columns[passed]]

            if log_transform:
                self.data = np.log(1+self.data)
                mean_filt = self.data.apply(np.mean) > mean_filter
                self.data = self.data.loc[:,mean_filt]

            if sc == "z-score":
                self.scaler = StandardScaler(with_mean = with_mean, with_std = with_std).fit(self.data)
            elif sc == "robust":
                self.scaler = RobustScaler(with_centering = with_mean, with_scaling = with_std).fit(self.data)
            elif sc == "power":
                self.scaler = PowerTransformer().fit(self.data)

            if self.scaler:
                self.data = pd.DataFrame(self.scaler.transform(self.data), columns = self.data.columns)

    def __getitem__(self, key):
        return self.data[key]

class Metabolite(Base):
    def __init__(self, filename = None, lowpass = True, threshold = 0.1, sc = "z-score", with_mean = True, with_std = True):
        super().__init__(filename, lowpass, threshold, sc, with_mean, with_std)

class Immune(Base):
    def __init__(self, filename = None, lowpass = True, threshold = 0.1, sc = "z-score", with_mean = True, with_std = True):
        super().__init__(filename, lowpass, threshold, sc, with_mean, with_std)

class Gene(Base):
    def __init__(self, filename = None, lowpass = True, threshold = 0.1, sc = "z-score", with_mean = True, with_std = True, log_transform = True, mean_filter = 2):
        super().__init__(filename, lowpass, threshold, sc, with_mean, with_std, log_transform, mean_filter)

class Data:
    def __init__(self, X_train, y_train, X_test = None, y_test = None, seed = 42):
        if X_test and y_test:
            self.common_features = set(X_test.data.columns)
            self.common_targets = set(y_test.data.columns)
        else:
            self.common_features = set(X_train.data.columns)
            self.common_targets = set(y_train.data.columns)

        if type(X_train) == list and type(y_train) == list:
            for X_, y_ in zip(X_train, y_train):
                self.common_features = set(X_.data.columns).intersection(self.common_features)
                self.common_targets = set(y_.data.columns).intersection(self.common_targets)
        else:
            self.common_features = set(X_train.data.columns).intersection(self.common_features)
            self.common_targets = set(y_train.data.columns).intersection(self.common_targets)

        self.common_features = list(self.common_features)
        self.common_targets = list(self.common_targets)

        if type(X_train) == list and type(y_train) == list:
            X = [x.data[self.common_features] for x in X_train]
            y = [y.data[self.common_targets] for y in y_train]

            self.X_train = pd.concat(X)
            self.y_train = pd.concat(y)
        else:
            self.X_train = X_train.data[self.common_features]
            self.y_train = y_train.data[self.common_targets]

        if X_test and y_test:
            self.X_test = X_test.data[self.common_features]
            self.y_test = y_test.data[self.common_targets]
        else:
            self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(self.X_train, self.y_train, test_size = 0.33, random_state = seed)

        str_to_replace = []#["[","]","<", ">"]
        for s in str_to_replace:
            self.X_train.columns = self.X_train.columns.str.replace(s,"")
            self.y_train.columns = self.y_train.columns.str.replace(s,"")
            self.X_test.columns = self.X_test.columns.str.replace(s,"")
            self.y_test.columns = self.y_test.columns.str.replace(s,"")

    def add(self, train_metadata, test_metadata):
        if train_metadata.shape[0] != self.X_train.shape[0]:
            print("number of training sample does not match")
            return
        if test_metadata.shape[0] != self.X_test.shape[0]:
            print("number of test sample does not match")
            return
        if train_metadata.shape[1] != test_metadata.shape[1]:
                print("number of training column and test column does not match")
                return
            self.X_train = pd.concat([self.X_train, train_metadata], axis=1)
            self.X_test = pd.concat([self.X_test, test_metadata], axis = 1)
```

Spearmanr.py

```python
from scipy.stats import spearmanr, pearsonr
from statsmodels.stats.multitest import multipletests
import pandas as pd
from tqdm import tqdm

class Correlation:
    def __init__(self, X = None, y = None, filename = None, corr_type = "spearman"):
        # if a filename is given, load the file
        if filename:
            self.data = pd.read_csv(filename)
        # otherwise compute correlations
        else:
            if X.empty or y.empty:
                print("invalid argument")
                return
            result = []
            for target in tqdm(y.columns):
                # batch = []
                for feature in X.columns:
                    if corr_type == "spearman":
                        r, pval = spearmanr(X[feature], y[target])
                    elif corr_type == "pearson":
                        r, pval = pearsonr(X[feature], y[target])
                    item = tuple([feature, target, r, pval])
                    result.append(item)

            columns = ["X","y","R", "pval"]
            self.data = pd.DataFrame(result, columns = columns)
        self.data["X"] = self.data["X"].astype(str)
        self.data["y"] = self.data["y"].astype(str)


        str_to_replace = ["[","]","<", ">"]
        for s in str_to_replace:
            self.data["X"] = self.data["X"].str.replace(s,"")
            self.data["y"] = self.data["y"].str.replace(s,"")

    # compute adjusted pvalue based on benjamini_hochberg algorithm
    def adjusted_pval(self):
          def benjamini_hochberg(corr):
              result = multipletests(corr["pval"], method = "fdr_bh")
              corr["reject"] = result[0]
              corr["qval"] = result[1]
              return corr
          #self.data = self.data.groupby(by = "X").apply(benjamini_hochberg)
          self.data = benjamini_hochberg(self.data)

    def __getitem__(self, target):
        return self.data.loc[self.data["y"] == target]

    def save(self, filename):
        self.data.to_csv(filename, index = False)
```

Preprocessor.py

```python
import pandas as pd
import os

MASTERFILE = "MasterMapping_MetImmune.csv"
master = pd.read_csv(MASTERFILE, index_col=0)
datasets = master["Identifier"].unique()

for x in datasets:
    print(x)
    # process RC data
    cur_data = master[(master["Identifier"] == x) &  (master["Histology"] == "Clear Cell")]
    if not cur_data.any().any():
        continue
    order = cur_data[cur_data.columns[:3]]

    metfile = 'Metabolites_Processed_Immune/' + cur_data["MetabFile"].unique()[0]
    rnafile = 'Kuo_RNA_Processed/' + cur_data["RNAFile"].unique()[0]
    ithfile = 'Kuo_Processed_Deconvolution/' + cur_data["ITHFile"].unique()[0]

    try:
        cur_met = pd.read_csv(metfile, index_col = 0).T
        cur_rna = pd.read_csv(rnafile, index_col = 0).T
        cur_ith = pd.read_csv(ithfile, index_col = 0)
    except FileNotFoundError:
        print("one of ", metfile, rnafile, ithfile, "doesnt exist")
        continue

    cur_met["MetabID"] = cur_met.index
    cur_met.reset_index(inplace = True)
    cur_rna["RNAID"] = cur_rna.index
    cur_rna.reset_index(inplace = True)
    cur_ith["ITHID"] = cur_ith.index
    cur_ith.reset_index(inplace = True)

    tmp = order.merge(cur_met, how = "inner", on = "MetabID")
    tmp_order = tmp[tmp.columns[:3]]
    tmp = tmp_order.merge(cur_rna, how = "inner", on = "RNAID")
    tmp_order = tmp[tmp.columns[:3]]
    tmp = tmp_order.merge(cur_ith, how = "inner", on = "ITHID")
    tmp_order = tmp[tmp.columns[:3]]

    final_met = tmp_order.merge(cur_met, how = "inner", on = "MetabID")
    final_rna = tmp_order.merge(cur_rna, how = "inner", on = "RNAID")
    final_ith = tmp_order.merge(cur_ith, how = "inner", on = "ITHID")

    dir_ = 'normal/'
    directory = os.path.dirname("".join(["data/" + dir_, x, "/"]))

    if not os.path.exists(directory):
        os.makedirs(directory)

    final_met[final_met.columns[4:]].to_csv("data/" + dir_ + x + "/met.csv", index = False)
    final_rna[final_rna.columns[4:]].to_csv("data/" + dir_ + x + "/rna.csv", index = False)
    final_ith[final_ith.columns[4:]].to_csv("data/" + dir_ + x + "/ith.csv", index = False
)
```

Modelhub.py

```python
from tqdm import tqdm
import numpy as np
import pandas as pd

from sklearn.linear_model import LassoCV
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
from sklearn.svm import SVR

class ModelHub:
    def __init__(self, model = "LASSO"):
        self.model = model
        # full model or partial model
        self.models = dict()
        self.results = dict()
        if model in ("LASSO", "SVM"):
            self.coefs = dict()
        elif model in ("XGBoost", "RandomForest"):
            self.featimp = dict()

    def __call__(self, data, sig_corr = None, cv = 3, seed = 42, iter = 1e4, metadata = None, num_feature = None):
        #y_cols = data.y_train.columns
        #data.y_train.columns = [str(x) for x in range(len(y_cols))]

        for target_feature in tqdm(data.y_train.columns):
            # runs full mode if sig_corr is not given as a parameter
            if sig_corr:
                if num_feature:
                    features = sig_corr[target_feature].sort_values("qval")[:num_feature]["X"]
                    if len(features) == 0:
                        continue
                    #print(features)
                else:
                    features = list(set(sig_corr[target_feature]["X"]).intersection(set(data.X_train.columns)))
                if metadata:
                    features += metadata
                # if no correlated features exist, skip the current loop
                if not num_feature and not features:
                    continue
            else:
                features = data.X_train.columns

            if self.model == "LASSO":
                model = LassoCV(cv = cv, random_state = seed, max_iter = iter)
            elif self.model == "XGBoost":
                model = xgb.XGBRegressor(objective ='reg:squarederror', scoring='r2', random_state = seed)
            elif self.model == "RandomForest":
                model = RandomForestRegressor()
            elif self.model == "SVM":
                model = SVR(kernel = "linear")

            # train
            model.fit(data.X_train[features], np.ravel(data.y_train[target_feature]))

            # test
            train_score = model.score(data.X_train[features], data.y_train[target_feature])
            test_score = model.score(data.X_test[features], data.y_test[target_feature])

            # store everything
            self.models[target_feature] = model
            self.results[target_feature] = (train_score, test_score)
            if self.model in ("LASSO", "SVM"):
                self.coefs[target_feature] = pd.DataFrame({'X':features, 'Y': target_feature, 'coef':model.coef_})

            elif self.model in ("XGBoost", "RandomForest"):
                self.featimp[target_feature] = pd.DataFrame({'X':features, 'Y': target_feature, 'featimp':model.feature_importances_})

        self.results = pd.DataFrame(self.results).T.rename(columns = {0:"R2 train", 1:"R2 test"})

        if self.model in ("LASSO", "SVM"):
            self.coefs = pd.concat(self.coefs.values())

        elif self.model in ("XGBoost", "RandomForest"):
            self.featimp = pd.concat(self.featimp.values())

        return self.results

    def predict(self, data, sig_corr = None, num_feature = None, metadata = None):
        results = []
        for target_feature in tqdm(data.y_train.columns):
            if sig_corr:
                if num_feature:
                    features = sig_corr[target_feature].sort_values("qval")[:num_feature]["X"]
                    if len(features) == 0:
                        continue
                    #print(features)
                else:
                    features = list(set(sig_corr[target_feature]["X"]).intersection(set(data.X_train.columns)))
                if metadata:
                    features += metadata
                # if no correlated features exist, skip the current loop
                if not num_feature and not features:
                    continue
            else:
                features = data.X_train.columns

            train_pred = self.models[target_feature].predict(data.X_train[features])
            test_pred = self.models[target_feature].predict(data.X_test[features])

            train_score = self.models[target_feature].score(data.X_train[features], data.y_train[target_feature])
            test_score = self.models[target_feature].score(data.X_test[features], data.y_test[target_feature])

            train = pd.DataFrame({"Y": target_feature,
                                  "type" : "train",
                                  "label": data.y_train[target_feature],
                                  "pred": train_pred,
                                  "r2": train_score,
                                  "residual":(data.y_train[target_feature] - train_pred)**2})
            test = pd.DataFrame({"Y": target_feature,
                                 "type": "test",
                                 "label": data.y_test[target_feature],
                                 "pred": test_pred,
                                 "r2": test_score,
                                 "residual":(data.y_test[target_feature] - test_pred)**2})

            results.append(train.append(test))
            df = pd.concat(results)

            return df.sort_values(by = "r2", ascending = False)
```

Metith.py

```python
import pandas as pd
import numpy as np
import copy

from codes.spearmanr import Correlation
from codes.loader import Metabolite, Immune, Data
from codes.modelhub import ModelHub

from collections import defaultdict

import matplotlib.pyplot as plt

SCALE = "z-score"

DIRECTORY_FILENAME = "data/directories.csv"
CORR_FILENAME = "results/rna/RC/corr2/allcorr.csv"

'''''
si18mr = pd.read_csv("data/RC18MR/si.csv")
si18fc = pd.read_csv("data/RC18Flow/si.csv")
si12 = pd.read_csv("data/RC12/si.csv")
'''


if __name__ == "__main__":

    # read in directories
    directories = pd.read_csv(DIRECTORY_FILENAME)
    # read in filenames from the directory
    #directories = directories[directories["type"] == "RC"]

    #load all files
    met = dict()
    ith = dict()
    corr = dict()
    train_test_split = defaultdict(lambda:[[]], dict())

    for idx, row in directories.iterrows():
        dir_ = 'data/' + row['status'] + '/' + row['data'] + '/'
        mfile = dir_ + 'met.csv'
        ifile = dir_ + 'ith.csv'
        cfile = dir_ + 'corr_metith.csv'

        met[row["data"]] = Metabolite(mfile, sc=SCALE)
        ith[row["data"]] = Immune(ifile, sc=SCALE)
    data = Data(X_train = [met["RC18MR"], met["RC12"]], y_train = [ith["RC18MR"], ith["RC1
2"]], X_test = met["RC18Flow"], y_test = ith["RC18Flow"])

    #corr["RC"] = Correlation(filename = "data/corr/RCcorr_metith.csv")
    #corr["BRCA"] = Correlation(filename = "data/corr/BRCAcorr_metith.csv")
    #corr["PRAD"] = Correlation(filename = "data/corr/PRADcorr_metith.csv")
    full_lasso = ModelHub(model = "LASSO")
    full_lasso(data = data)
    print(full_lasso.results[(full_lasso.results > 0.2).all(axis = 1)].sort_values("R2 tes
t", ascending = False))

```

```python
    cr = Correlation(data.X_train, data.y_train)
    pd.pivot(cr.data, index = "X", columns = "y", values = "R").to_csv("metith/corr.csv")

    pd.pivot(full_lasso.coefs, index = "X", columns = "Y", values="coef").to_csv("metith/lasso_coef.csv")

    partial_rf = ModelHub(model = "RandomForest")
    partial_rf(data = data)
    print(partial_rf.results[(partial_rf.results > 0.2).all(axis = 1)].sort_values("R2 test", ascending = False))


    partial_xgb = ModelHub(model = "XGBoost")
    partial_xgb(data = data)
    print(partial_xgb.results[(partial_xgb.results > 0.2).all(axis = 1)].sort_values("R2 test", ascending = False))


    full_lasso.predict(data).to_csv("metith/lasso_residuals.csv", index = False)
    partial_rf.predict(data).to_csv("metith/rf_residuals.csv", index = False)
    partial_xgb.predict(data).to_csv("metith/xgb_residuals.csv", index = False)

    full_lasso.results["Y"] = full_lasso.results.index()
```

Rnaith.py

```python
import pandas as pd
import numpy as np
import copy

from codes.spearmanr import Correlation
from codes.loader import Metabolite, Gene, Immune, Data
from codes.modelhub import ModelHub
from collections import defaultdict

from sklearn.linear_model import LassoCV

import matplotlib.pyplot as plt
from tqdm import tqdm
from seaborn import heatmap
from sklearn.cross_decomposition import PLSRegression
from sklearn.metrics import r2_score

SCALE = "z-score"
STATUS = "tumor"
DIRECTORY_FILENAME = "data/directories.csv"
# CORR_FILENAME = "results/rna/RC/corr2/allcorr.csv"

if __name__ == "__main__":
    # read in directories
    directories = pd.read_csv(DIRECTORY_FILENAME)
    # read in filenames from the directory
    directories = directories[(directories["status"] == STATUS)] # & (directories["type"]
!= "RC")]

    #load all files
    met = dict()
    ith = dict()
    gene = dict()
    corr = dict()
    data = dict()
    train_test_split = defaultdict(lambda:[], dict())

    for idx, row in directories.iterrows():
        dir_ = 'data/' + row['status'] + '/' + row['data'] + '/'
        mfile = dir_ + 'met.csv'
        ifile = dir_ + 'ith.csv'
        gfile = dir_ + 'rna.csv'
        cfile = dir_ + 'corr_rnamet.csv'

        met[row["data"]] = Metabolite(mfile, sc=SCALE)
        ith[row["data"]] = Immune(ifile, sc=SCALE)
        gene[row["data"]] = Gene(gfile, sc=SCALE)
        try:
            corr[row["data"]] = Correlation(filename = cfile)
        except FileNotFoundError:
            print("Correlation file not found.")
            continue

    for key in gene:
        print(key, str(ith[key].data.shape))

    # train test split
    # data = Data(X_train = [gene["RC18MR"], gene["RC12"]], y_train = [met["RC18MR"], met["RC12"]], X_test = gene["RC18Flow"], y_test = met["RC18Flow"])

    for idx, df in directories.groupby(["type"]):
        if type(df) is pd.DataFrame:
            train_data = df[df["set"] == "train"]["data"].tolist()
            test_data = df[df["set"] == "test"]["data"].tolist()
            dataset = df["type"].unique()[0]

            if train_data and test_data:
                if len(train_data) > 1:
                    xi = [gene[x] for x in train_data]
                    yi = [met[x] for x in train_data]
                else:
                    xi = gene[train_data[0]]
                    yi = met[train_data[0]]
                if len(test_data) > 1:
                    xo = [gene[x] for x in test_data]
                    yo = [met[x] for x in test_data]
                else:
                    xo = gene[test_data[0]]
                    yo = met[test_data[0]]
                data[dataset] = Data(X_train = xi, y_train = yi, X_test = xo, y_test = yo)

            else:
                data[dataset] = Data(X_train = gene[dataset], y_train = met[dataset])


    lasso_modelhub = dict()
    rf_modelhub = dict()
    cancer_results = dict()

    pls = PLSRegression(n_components=4)
    pls.fit(data['RC'].X_train, data['RC'].y_train[metabolites])
    #pls.score(data['RC'].X_train, data['RC'].y_train[metabolites])
    #pls.score(data['RC'].X_test, data['RC'].y_test[metabolites])
    y = pd.DataFrame(pls.predict(data['RC'].X_test), columns = metabolites)
    pls_result = dict()
    for key in y:
        pls_result[key] = r2_score(data['RC'].y_test[key], y[key])
    pls_result = pd.Series(pls_result)
    print(pls_result)

    plt.plot(y[:,0],data['RC'].y_test[metabolites[0]],'.')

    metabolites = full_lasso.results.sort_values('R2 test', ascending = False).head(5).index

    for key in data:
        key = 'RC'
        print(key)
        full_lasso = ModelHub(model = "LASSO")
        full_lasso(data = data[key])
        full_lasso.results.columns = 'L1 ' + full_lasso.results.columns

        rf_model = ModelHub(model = "RandomForest")
        rf_model(data = data[key])
        rf_model.results.columns = 'RF ' + rf_model.results.columns

```

```
            combined_result = full_lasso.coefs\
                .merge(rf_model.featimp, how = 'left', on = ['X','Y'])\
                .merge(full_lasso.results, how = 'left', left_on = 'Y', right_index = True)\
                .merge(rf_model.results, how = 'left', left_on = 'Y', right_index = True)

            important_results = combined_result[(combined_result['coef'].abs()> 0) | (combined_result['featimp'] > 0)]

            important_results['Importance'] = important_results['L1 R2 test'] * \
                np.exp(important_results['coef'].abs() * 10) + \
                important_results['RF R2 test'] * \
                np.exp(important_results['featimp'] * 10)

            important_results = important_results.sort_values('Importance', ascending = False)
            important_results.to_csv(key +' important results.csv', index = False)

            lasso_modelhub[key] = full_lasso
            rf_modelhub[key] = rf_model
            cancer_results[key] = important_results

        tmp = cancer_results
        for key in cancer_results:
            tmp[key].set_index(['X','Y'], inplace = True)

        result = pd.concat(tmp,axis=1)
        result.reset_index(inplace = True)
        result.columns = ['X','Y'] + result.columns.get_level_values(1)[2:].tolist()
        result['Mean Importance'] = result.loc[:,result.columns.str.contains('Importance')].mean(axis=1)
        new_col_order = result.columns[:2].tolist() + [result.columns[-1]] + result.columns[2:-1].tolist()
        result = result[new_col_order].sort_values('Mean Importance', ascending = False)

        result.to_csv('pancancer result.csv',index = False)

            tmp[key].columns = cancer_results[key].columns[:2] \
                .append(key + ' ' + cancer_results[key].columns[2:])
            '''''
            print(full_lasso.results[(full_lasso.results > 0.1).all(axis = 1)].sort_values("R2 test", ascending = False))

            lasso_modelhub[key] = full_lasso
            full_lasso.predict(data[key]).to_csv("rnamet/"+ key + "lasso_residuals.csv", index = False)
            pd.pivot(full_lasso.coefs, index = "X", columns = "Y", values="coef").to_csv("rnamet/" + key + "lasso_coef.csv")
            '''

        results = []
        for key in lasso_modelhub:
            tmp_result = lasso_modelhub[key].results.sort_values(by = "R2 test", ascending = False)
            tmp_result["type"] = key
            results.append(tmp_result)

        pd.concat(results).sort_values(by= "R2 test", ascending = False).to_csv("final result.csv")

        '''''
        full_rf = ModelHub(model = "RandomForest")
        full_rf(data = data)
        print(full_rf.results[(full_rf.results > 0).all(axis = 1)])

        full_rf.predict(data).to_csv("rnamet/rf_residuals.csv", index = False)

        full_xgb = ModelHub(model = "XGBoost")
        full_xgb(data = data)
        print(full_xgb.results[(full_xgb.results > 0).all(axis = 1)])

        full_xgb.predict(data).to_csv("rnamet/xgb_residuals.csv", index = False)
        '''

        coefs = dict()
        corrs = dict()
        results = dict()
        r2_results = dict()

        N = 30
        for key in lasso_modelhub:
            coefs[key] = pd.pivot(lasso_modelhub[key].coefs, index = "X", columns = "Y", values="coef")
            #corrs[key] = pd.pivot(corr[key].data, index = "X", columns = "y", values="R")

            lasso_modelhub[key].results = lasso_modelhub[key].results.sort_values("R2 test", ascending = False)
            filtdf = lasso_modelhub[key].results[lasso_modelhub[key].results["R2 test"] > 0][:N]
            mets = filtdf.index
            genes = coefs[key].loc[:,mets].abs().idxmax().unique()

            #genes1 = corrs[key].index.intersection(genes)
            #mets1 = corrs[key].columns.intersection(mets)
            r2_results[key] = filtdf
            results[key] = coefs[key].loc[genes, mets]

            results[key].to_csv("rnamet/" + key + "coef_heatmap.csv")
            r2_results[key].to_csv("rnamet/" + key + "r2_heatmap_annotation.csv")


        tmp_heatmap_data = pd.concat(results.values(), sort = False)
        rows = tmp_heatmap_data.index
        cols = tmp_heatmap_data.columns

        coef_data_dict = dict()
        r2 = dict()
        for key in coefs:
            tmp_coef_data = pd.DataFrame(np.NaN, index = rows, columns = cols)

            c_rows = coefs[key].index.intersection(rows)
            c_cols = coefs[key].columns.intersection(cols)

            r2[key] = lasso_modelhub[key].results.loc[c_cols,"R2 test"]

            tmp_coef_data.loc[c_rows, c_cols] = coefs[key].loc[c_rows, c_cols]
            coef_data_dict[key] = tmp_coef_data

        r2df = pd.DataFrame(r2)
        heatmap_data = coef_data_dict['BRCA']
        for key in coef_data_dict:
            heatmap_data = heatmap_data.where(heatmap_data.abs() > coef_data_dict[key].abs(
), coef_data_dict[key]).fillna(heatmap_data)

        coef_data_dict['BRCA'].abs().max(axis=1) > 0
        col_ann_data = dict()
        for key in coef_data_dict:
            col_ann_data[key] = coef_data_dict[key].abs().max(axis=1)

        colAnn = pd.DataFrame(col_ann_data)

        heatmap_data.T.to_csv("rnamet/heatmap.csv")
        colAnn.idxmax(axis=1).to_csv("rnamet/colann.csv")
        r2df.loc[heatmap_data.columns].to_csv("rnamet/rowann.csv")
        heatmap(heatmap_data.T, cmap = 'RdBu_r')

        ds = 'RC'
        g = '3620'#'1347'
        m = 'kynurenine'#'histamine'
        plt.figure(figsize=(10,6), dpi = 100)
        plt.plot(data[ds].X_train[g], data[ds].y_train[m], '.', label = 'Train')
        plt.plot(data[ds].X_test[g], data[ds].y_test[m], '.', label = 'Test')
        plt.plot(np.linspace(-2,2),np.linspace(-2,2),'r--')
        plt.xlabel("IDO1 Level")
        plt.ylabel("kynurenine Level")
        plt.legend(loc = "lower right")
        plt.savefig("figure1.png")
```

References

[1]. Barbie, D. A., Tamayo, P., Boehm, J. S., Kim, S. Y., Moody, S. E., Dunn, I. F., … Hahn, W. C. (2009, November 5). Systematic RNA interference reveals that oncogenic KRAS-driven cancers require TBK1. Retrieved April 7, 2020, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2783335/

[2]. Case Studies in Immunotherapy. (n.d.). Retrieved April 7, 2020, from http://www.personalizedmedonc.com/publications/ito/september-2014-part-3/case-studies-in-immunotherapy/

[3]. Chen, Tianqi, & Carlos. (2016, June 10). XGBoost: A Scalable Tree Boosting System. Retrieved April 12, 2020, from https://arxiv.org/abs/1603.02754

[4]. Chuah, S., & Chew, V. (2020, February 1). High-dimensional immune-profiling in cancer: implications for immunotherapy. Retrieved April 12, 2020, from https://jitc.bmj.com/content/8/1/e000363

[5]. Decision Tree Regression¶. (n.d.). Retrieved April 7, 2020, from https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html

[6]. SPRINGER. (2020). *Elements Of Statistical Learning*. S.l.

[7]. Fridman, W. H., Galon, J., Dieu-Nosjean, M.-C., Cremer, I., Fisson, S., Damotte, D., … Sautès-Fridman, C. (2011). Immune infiltration in human cancer: prognostic significance and disease control. Retrieved April 12, 2020, from https://www.ncbi.nlm.nih.gov/pubmed/20512556

[8]. How genes can help in the diagnosis and treatment of cancer. (n.d.). Retrieved April 7, 2020, from https://www.cancer.org/cancer/cancer-causes/genetics/genes-and-cancer/genes-in-cancer-diagnosis-and-treatment.html

[9]. Immunotherapy by Cancer Type. (n.d.). Retrieved April 7, 2020, from https://www.cancerresearch.org/immunotherapy/cancer-types

[10]. Immunotherapy for Cancer. (n.d.). Retrieved April 12, 2020, from https://www.cancer.gov/about-cancer/treatment/types/immunotherapy

[11]. Kingsford, C., & Salzberg, S. L. (2008, September). What are decision trees? Retrieved April 12, 2020, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2701298/

[12]. Labadie, B. W., Bao, R., & Luke, J. J. (2019, March 1). Reimagining IDO Pathway Inhibition in Cancer Immunotherapy via Downstream Focus on the Tryptophan–Kynurenine–Aryl Hydrocarbon Axis. Retrieved April 7, 2020, from https://clincancerres.aacrjournals.org/content/25/5/1462

[13]. Liu, M., Wang, X., Wang, L., Ma, X., Gong, Z., Zhang, S., & Li, Y. (2018, August 2). Targeting the IDO1 pathway in cancer: from bench to bedside. Retrieved April 7, 2020, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6090955/

[14]. Lorraine. (n.d.). Classification and Regression Analysis with Decision Trees. Retrieved April 7, 2020, from https://dev.to/nexttech/classification-and-regression-analysis-with-decision-trees-jgp

[15]. Potter, M. (2017, July 18). Anti-PD-1: A Novel Immunotherapy: Johns Hopkins Kimmel Cancer Center. Retrieved April 12, 2020, from https://www.hopkinsmedicine.org/kimmel_cancer_center/centers/melanoma/research/anti_pd_1.html

[16]. Priolo, C., Khabibullin, D., Reznik, E., Ogórek, B., Kavanagh, T. R., Nijmeh, J., … Henske, E. P. (2018, July 3). Impairment of gamma-glutamyl transferase 1 activity in the metabolic pathogenesis of chromophobe renal cell carcinoma. Retrieved April 7, 2020, from https://www.pnas.org/content/115/27/E6274

[17]. Regan, T., Gill, A. C., Clohisey, S. M., Barnett, M. W., Pariante, C. M., Harrison, N. A., … Freeman, T. C. (2018, April). Effects of anti-inflammatory drugs on the expression of tryptophan-metabolism genes by human macrophages. Retrieved April 7, 2020, from https://www.ncbi.nlm.nih.gov/pubmed/29377288

[18]. RNA-Seq. (2020, April 2). Retrieved April 7, 2020, from https://en.wikipedia.org/wiki/RNA-Seq#/media/File:Summary_of_RNA-Seq.svg

[19]. Schrimpe-Rutledge, A. C., Codreanu, S. G., Sherrod, S. D., & McLean, J. A. (2016, December). Untargeted Metabolomics Strategies-Challenges and Emerging Directions. Retrieved April 12, 2020, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5110944/

[20]. Selwan, E. M., & Edinger, A. L. (2017, May). Branched chain amino acid metabolism and cancer: the importance of keeping things in context. Retrieved April 7, 2020, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6319930/

[21]. Sha, Y., Phan, J. H., & Wang, M. D. (2015). Effect of low-expression gene filtering on detection of differentially expressed genes in RNA-seq data. Retrieved April 12, 2020, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4983442/

[22]. sklearn.decomposition.PCA¶. (n.d.). Retrieved April 12, 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[23]. sklearn.feature_selection.VarianceThreshold¶. (n.d.). Retrieved April 12, 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html

[24]. Spearman's Rank-Order Correlation. (n.d.). Retrieved April 12, 2020, from https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide.php

[25]. What is Mass Spectrometry? (2016, June 24). Retrieved April 12, 2020, from https://www.broadinstitute.org/proteomics/what-mass-spectrometry

[26]. What is metabolomics? (2016, June 8). Retrieved April 7, 2020, from https://www.ebi.ac.uk/training/online/course/introduction-metabolomics/what-metabolomics

[27]. What is transcriptomics? (n.d.). Retrieved April 12, 2020, from https://www.phgfoundation.org/blog/what-is-transcriptomics

[28]. Why Immunotherapy? (n.d.). Retrieved April 7, 2020, from https://www.cancerresearch.org/immunotherapy/why-immunotherapy