Cancer Gene Expression Classification

SPRINGBOARD

Capstone Project 3

2023 | MAY

TAMARA HORNE

Table of Contents

Introduction

Medical providers and researchers can diagnose the cancer type of tumors more quickly and accurately with the assistance of machine learning. However, developing a meaningful machine learning method is difficult because of the curse of dimensionality. The available dataframes are regularly wide and short because of the vast number of genes and the relatively small number of samples. The dataframe I'll use in this project is no different.

The gene expression cancer RNA_Seq data (https://archive-beta.ics.uci.edu/dataset/401/gene+expression+cancer+rna+seq) is part of the larger RNA_Seq (HiSeq) PANCAN data set. (PANCAN is the Pancreatic Cancer Action Network). This database consists of samples representing 5 different cancer types. This project seeks to find a model which will accurately classify samples in all five classes.

Using six models, some linear and some non-linear, I achieved f1 macro scores ranging from 0.979-0.994. KNeighborsClassifier, Logistic Regression, and Support Vector Classification all tied for the highest score and would be excellent models for predicting future samples. Using SHAP with XGBoostClassifier, I was able to show which features were most important in predicting each cancer type. For additional details, please visit my GitHub repository folder: https://github.com/tamarahorne/Springboard/tree/main/Capstone%20Project%202.

Data Acquisition & Wrangling

The gene expression cancer RNA_Seq data consists of 801 rows representing samples and 20531 features. The features are RNA_seq gene expression levels which were measured using the illumina HiSeq platform. The data contained no missing values and their were no exceptionally high or low values in columns to indicate a mistake (figure 1); the maximum value in the dataframe was 20.778 and the minimum value was 0. So, I turned my attention to other concerns.

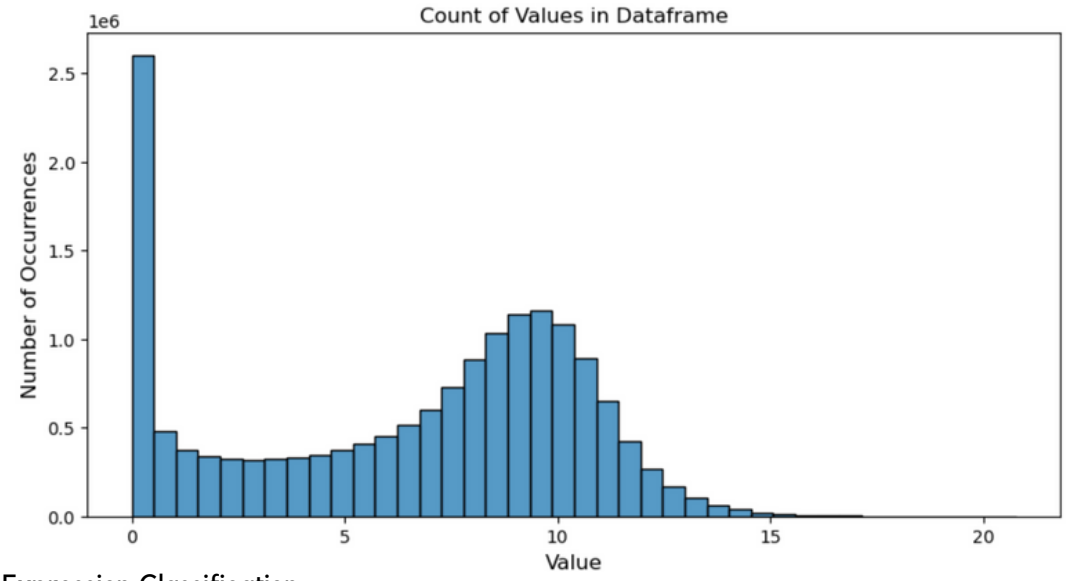
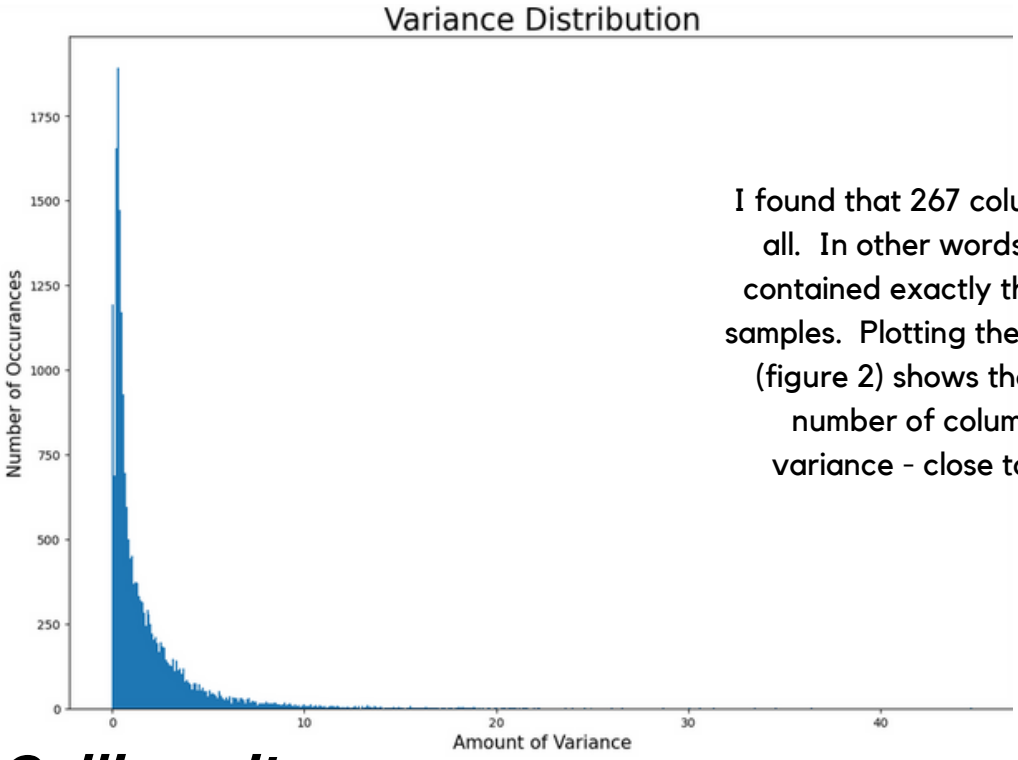


figure 1

Storytelling

With a wide and short dataframe, the first concern is going to be the number of dimensions. I wanted to gain a better understanding of the amount of variance within each column, the frequency of high collinearity between pairs of columns, and the balance of classes in order to choose an appropriate path for dimensional reduction and/or feature selection.



Variance

I found that 267 columns had no variance at all. In other words, each of those columns contained exactly the same value for all 801 samples. Plotting the distribution of variance (figure 2) shows that there are also a large number of columns which have very low variance - close to but not equal to zero.

figure 2

Collinearity

Next, I checked for collinearity between pairs of features (figure 3) since feature pairs with high collinearity cannot independently predict the value of the target. I discovered 2254 pairs of features with a correlation coefficient of over 90%. So, one way to the reduce the number of features could be to remove one feature from each of the highly collinear pairs. Six genes in the dataframe showed high collinearity with 50 or more genes each, and another 13 genes paired with 40 or more genes. So, these would be ideal candidates for removal.

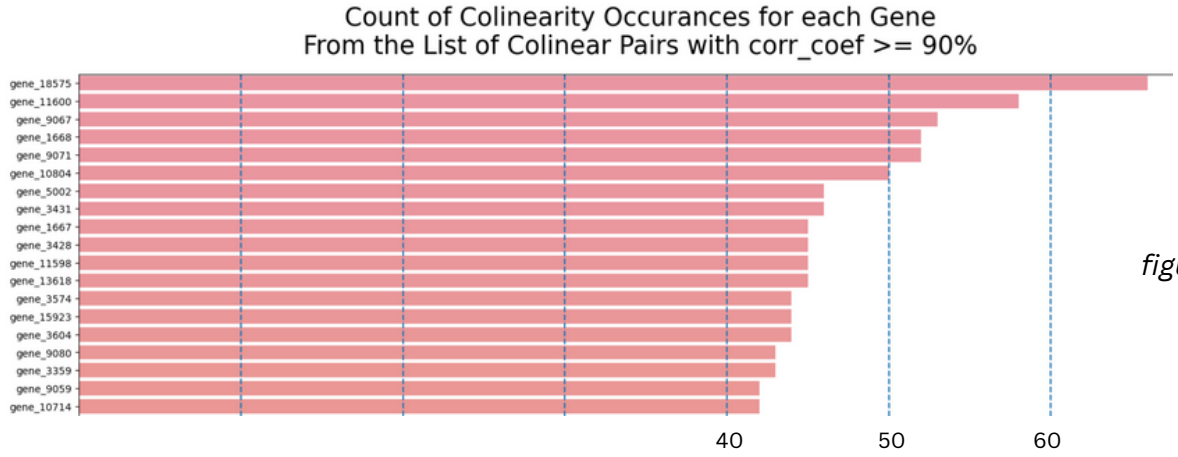


figure 3

Storytelling (continued)

Balance

The dataset has a class imbalance but it is not severe (figure 4). The majority class, Class 0: BRCA ((breast), accounts for 37% of the samples. The smallest minority class, Class 1: COAD (colon), accounts for 10% of the samples. Oversampling and undersampling are common ways to treat data imbalance, but both methods have a downside. Oversampling can cause overfitting because it works by duplicating existing data. Undersampling reduces the number of samples in the larger classes yielding a smaller number of total samples available to train and test the model.

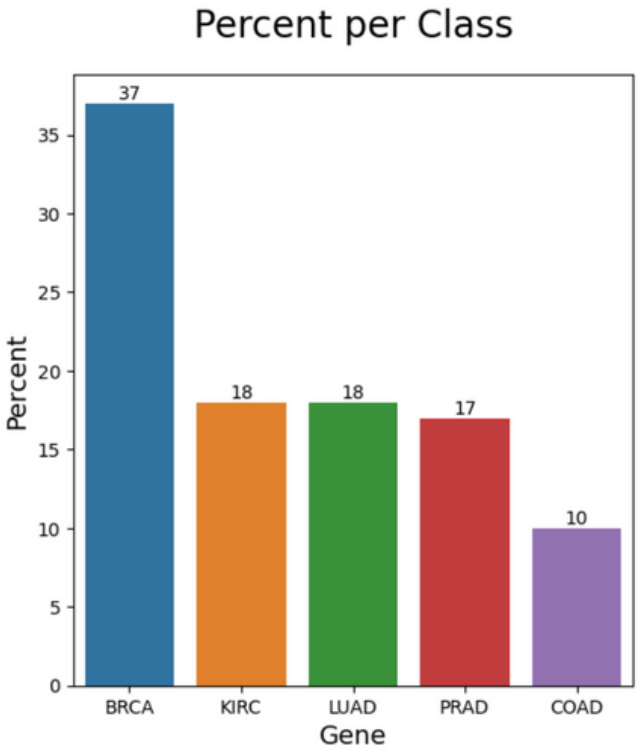


figure 4

Decisions

After considering the number of dimensions, the variance, and the collinearity, I decided PCA (principal component analysis) would be a good option to try. PCA takes all of those elements into consideration when choosing what comprises each principal components.

In regards to the data imbalance, I decided to try modeling without utilizing either method (oversampling or undersampling) to see how well the baseline model worked without such intervention. I stratified the data when I created the train/test split in order to preserve the ratio of samples per class.

Baseline Modeling

I used logistic regression for a baseline model. Before I could move forward with modeling, though, I needed to choose the landing dimension for PCA. For a multi-class classification problem, the rule of thumb is that a dataframe should have at least 10 samples per feature for the smallest class. In other words, there should be at least ten times more rows than columns. The dataframe I am using has 801 samples, but the smallest class only has 78. So, according to the rule of thumb, the number of features should be restricted to 7 or, if I round, 8.

Baseline Modeling (continued)

Plotting the individual and cumulative variance explained by the first seven principal components (figure 5) gave me confidence that following the rule of thumb would be sufficient for this project. Further, when we look at a plot of just the first two principal components (figure 6), it seems entirely likely an even smaller number of principal components would be sufficient. An extension of this project would be to fine tune the number of principal components by comparing computation time against model effectiveness.

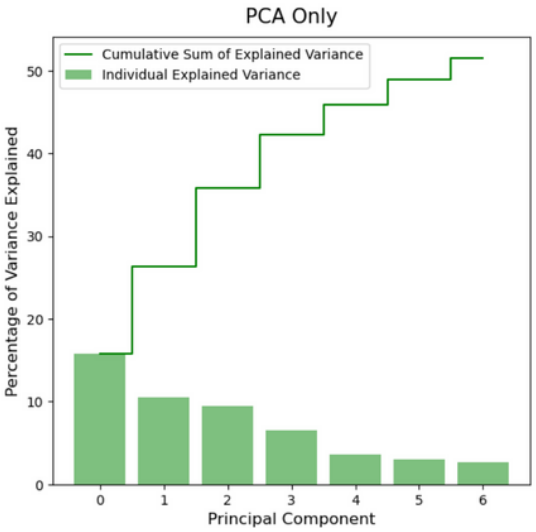


figure 5

Plot of First Two Principal Components

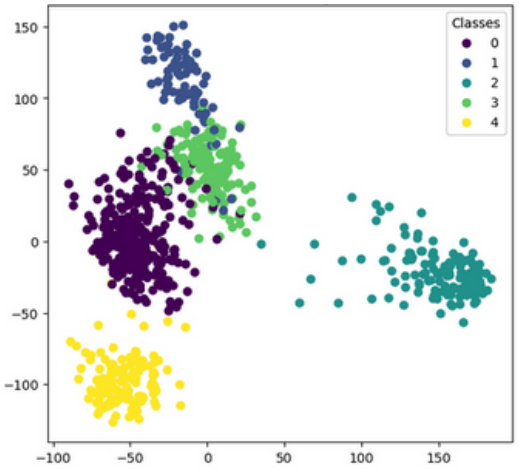


figure 6

I explored the idea of scaling the data, since that is a typical step to take before using PCA. However, I found that scaling made the amount of variance explained by the first seven principal components go down (figure 7), it made the classes less defined when plotting the first two principal components (figure 8), and the confusion matrix for the baseline linear regression model showed an increase in the number of errors (figure 9). So, I made the decision to proceed with unscaled data.

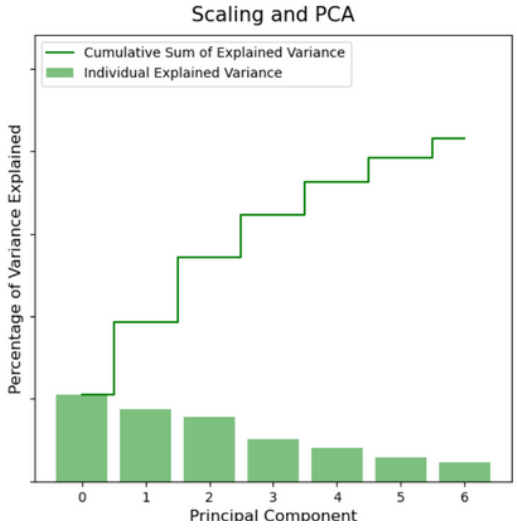


figure 7

Plot of First Two Principal Components

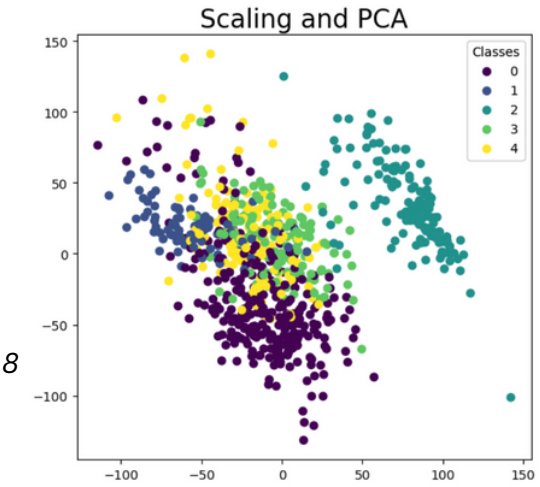


figure 8

Baseline Modeling (continued)

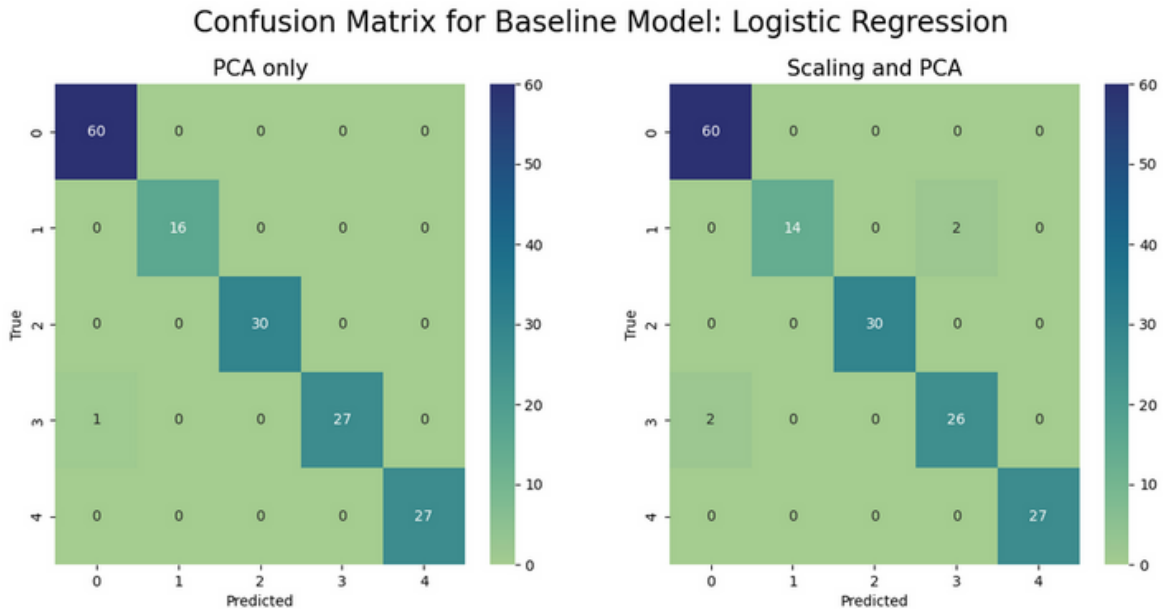


figure 9

The logistic regression model with PCA performs very well for all classes so further action to balance the classes is unnecessary. The success of the baseline models also confirms that the landing dimension is sufficient. I used KStratifiedFold, which preserves the class balance in each fold to validate the success of the model on the training data, just as an added check (figure 11). The scores were consistent enough to give me confidence the model would generalize well, and indeed it did as we can see below with the classification report for the testing data (figure 10).

Training Data

Train set classification report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	240
1	1.00	1.00	1.00	62
2	1.00	1.00	1.00	116
3	1.00	1.00	1.00	113
4	1.00	1.00	1.00	109
accuracy			1.00	640
macro avg	1.00	1.00	1.00	640
weighted avg	1.00	1.00	1.00	640

Testing Data

Test set classification report:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	60
1	1.00	1.00	1.00	16
2	1.00	1.00	1.00	30
3	1.00	0.96	0.98	28
4	1.00	1.00	1.00	27
accuracy			0.99	161
macro avg	1.00	0.99	0.99	161
weighted avg	0.99	0.99	0.99	161

figure 10

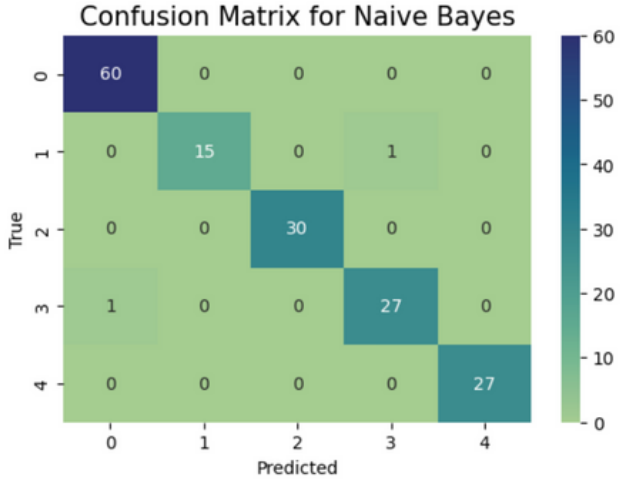
```
Fold number: 0, F1_macro_avg: 1.000
Fold number: 1, F1_macro_avg: 1.000
Fold number: 2, F1_macro_avg: 0.987
Fold number: 3, F1_macro_avg: 0.991
Fold number: 4, F1_macro_avg: 1.000
```

figure 11

Extended Modeling

After seeing the success of the logistic regression model, I was curious to see how other models, both linear and non-linear, would compare. I ran the linear models, Naive Bayes and Support Vector Classifier, in a pipeline with PCA. I then used StratifiedKFold for cross validation so the class balance would be preserved. Hyper-parameter tuning was not necessary since overfitting was not a concern. Finally, I collected the f1 macro score for each model and viewed the classification report and confusion matrix for each.

Naive Bayes - test data results



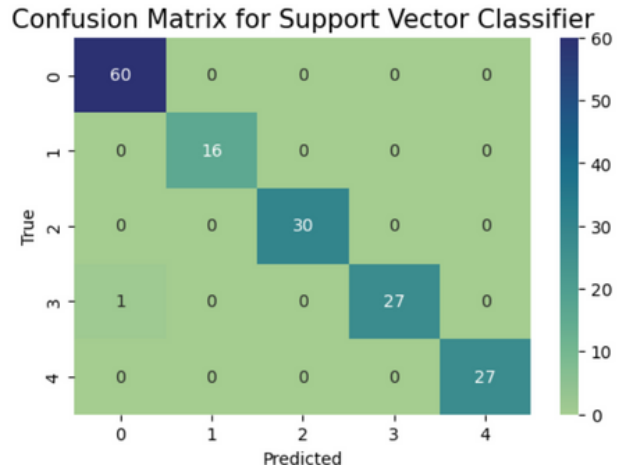
Naive Bayes classification report:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	60
1	1.00	0.94	0.97	16
2	1.00	1.00	1.00	30
3	0.96	0.96	0.96	28
4	1.00	1.00	1.00	27
accuracy			0.99	161
macro avg	0.99	0.98	0.98	161
weighted avg	0.99	0.99	0.99	161

figures 12 and 13

Like logistic regression, the Naive Bayes model makes one mistake for class three (figure 12), classifying it as class 0, but Naive Bayes makes one additional mistake - misclassifying one sample of class 1 as class 3. The classification report (figure 13) still shows very good results with an f1 macro score of 0.98.

Support Vector Classifier - test data results



Support Vector Classifier classification report:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	60
1	1.00	1.00	1.00	16
2	1.00	1.00	1.00	30
3	1.00	0.96	0.98	28
4	1.00	1.00	1.00	27
accuracy			0.99	161
macro avg	1.00	0.99	0.99	161
weighted avg	0.99	0.99	0.99	161

figures 14, and 15

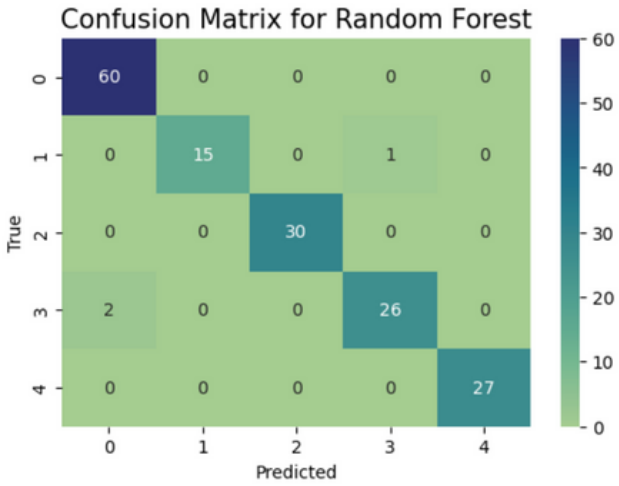
Support Vector Classifier mirrors logistic regression with the single misclassification of class 3 as class 0 (figure 14). The f1 macro average score is 0.99 (figure 15).

Extended Modeling (continued)

Next I tried a sampling of non-linear models including Random Forest Classifier, XGBoost Classifier, and K Neighbors Classifier. These models do require hyper-parameter tuning.

Random Forest Classifier

To find the best parameters for the Random Forest model, I tried a variety of values for the number of estimators, the max features, max depth, minimum sample leaf, minimum sample split, and the criterion. I then judged each combination by it's out-of-bag score (1 - oob_score). I then ran the model with the best parameters to find the f1 macro score of0.979 (figure 17). The confusion matrix shows three misclassified samples (figure 16).



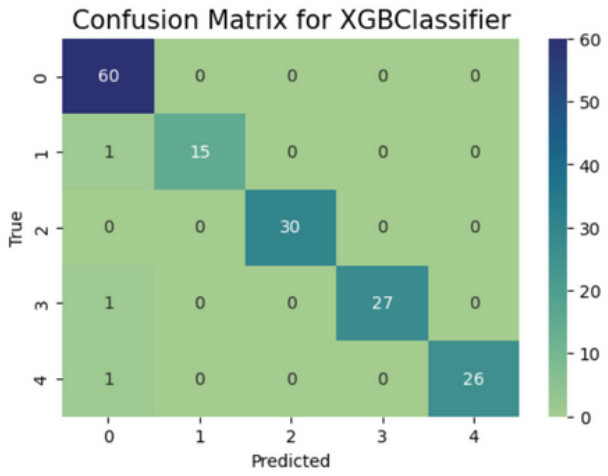
Random Forest classification report:

	precision	recall	f1-score	support
0	0.97	1.00	0.98	60
1	1.00	0.94	0.97	16
2	1.00	1.00	1.00	30
3	0.96	0.93	0.95	28
4	1.00	1.00	1.00	27
accuracy			0.98	161
macro avg	0.99	0.97	0.98	161
weighted avg	0.98	0.98	0.98	161

figures 16 and 17

XGBoost Classifier

I used hyper-parameter tuning again for the XGBoost Classifier but, since it does not have an out-of-bag error option, I had to go about it in a different way. I used early stopping with 'merror' as the evaluation metric, and I used the accuracy score as the measure for determining the best parameters. I then ran the model with the best parameters and found the f1 macro score to be 0.981 (figure 19). The confusion matrix shows a total of three errors - all of which are false positives for Class 0.



XGBClassifier classification report:

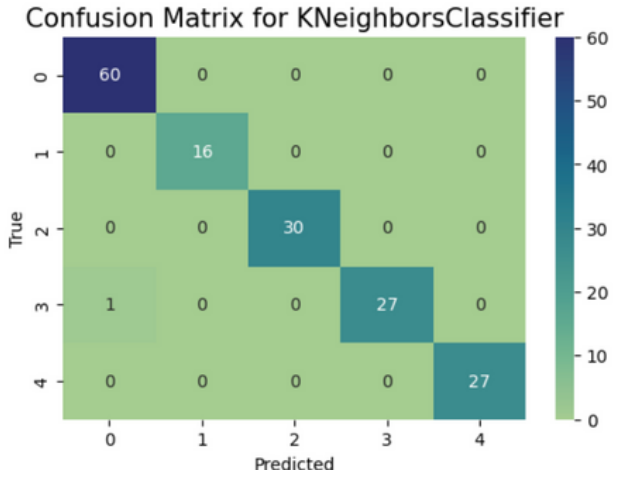
	precision	recall	f1-score	support
0	0.95	1.00	0.98	60
1	1.00	0.94	0.97	16
2	1.00	1.00	1.00	30
3	1.00	0.96	0.98	28
4	1.00	0.96	0.98	27
accuracy			0.98	161
macro avg	0.99	0.97	0.98	161
weighted avg	0.98	0.98	0.98	161

figures 18 and 19

Extended Modeling (continued)

K Neighbors Classifier

For the K Neighbors Classifier hyper-parameter tuning, I used Randomized Search CV to try a variety of values for the number of neighbors, the weights, the leaf size, and the value of p and scored the results using best_score. The model run with the best parameters yielded an f1 macro score of 0.9947 (figure 21) and only one misclassification (figure 20).



KNeighborsClassifier classification report:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	60
1	1.00	1.00	1.00	16
2	1.00	1.00	1.00	30
3	1.00	0.96	0.98	28
4	1.00	1.00	1.00	27
accuracy			0.99	161
macro avg	1.00	0.99	0.99	161
weighted avg	0.99	0.99	0.99	161

figures 20 and 21

Model Comparison

In the end, three models tied for the highest f1 macro score (figure 22). KNeighborsClassifier, Logistic Regression, and Support Vector Classifier all achieved a score of 0.994711. Moreover, the lowest f1 score of the whole group was only 0.01535 below the leaders. So, all six models performed very well.

model names	f1 scores
KNeighborsClassifier	0.994711
Logistic Regression	0.994711
Support Vector Classifier	0.994711
Naive Bayes	0.984753
XGBoostClassifier	0.981260
Random Forest	0.979361

figure 22

Findings

With all of the models performing so well, I was still left wanting to better understand why the were so predictable. We saw a taste of the answer with the plots of the first two principal components in figure 6. The multi-class plot of feature importance below*, though, begins to give us clarity (figure 23). Right away, we see each feature has exactly one color. meaning it is only registering as important for one class. Of the top 20 features in the dataframe, seven are important for predicting class 0, three are important for predicting class 1, six are important for predicting class 2, three for class 3, but only one feature is important for predicting class 4.

SHAP Summary Plot of Feature Importance Colored by Class

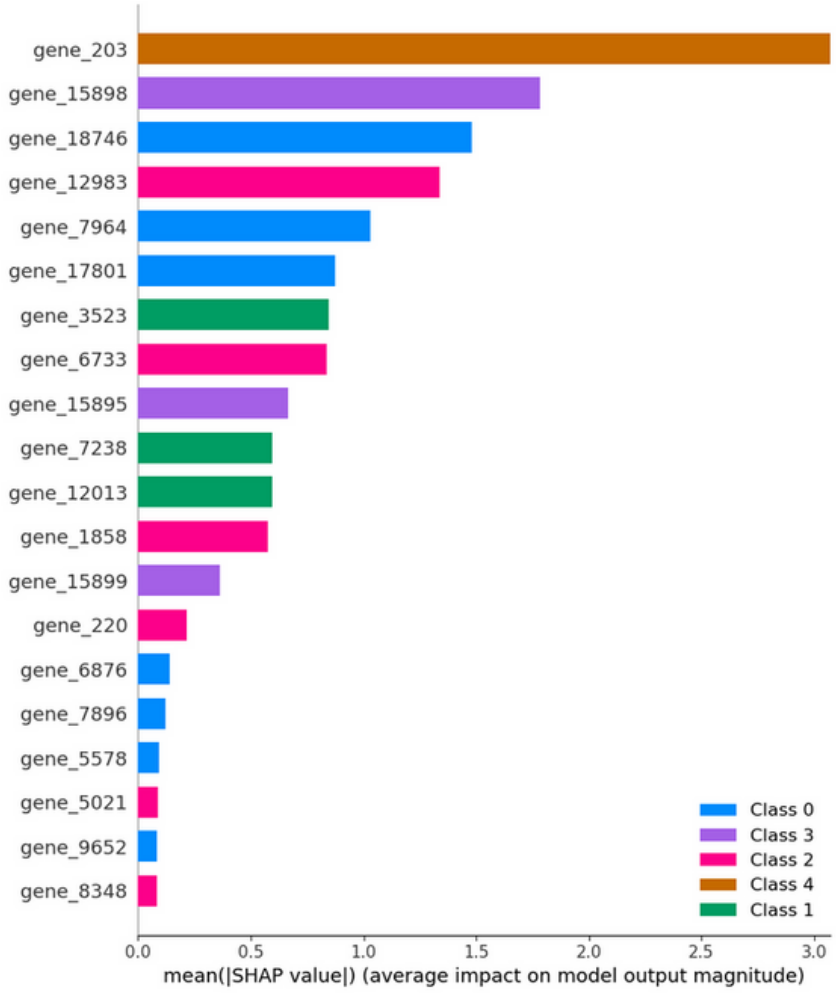


figure 23

Even when we examine the top 10 most important features for each class separately, we do not see any overlap of features between classes. (See figures 24, 29, 34, 39, and 45 in the *Findings* section of this report.)

**All SHAP visualizations in this document are based on the highest performing tree based model, XGBoostClassifier, so I could utilize SHAP's TreeExplainer. For further explanation for this choice, please visit https://github.com/tamarahorne/Springboard/blob/main/Capstone%20Project%203/notebooks/04_CGE_Expanded_Modeling.ipynb)*

Findings (continued)

Class 0: BRCA (breast)

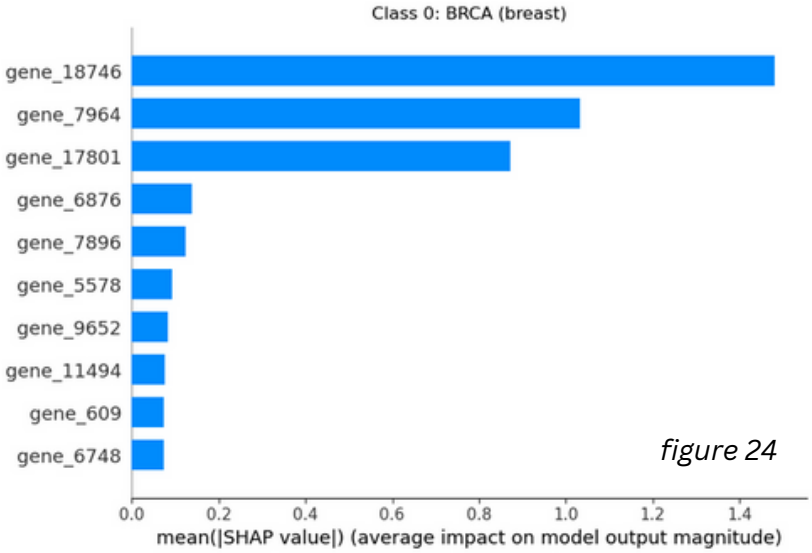


figure 24

Class 0 is the majority class and accounts for 37% of the samples. We can see from the feature importance plot (figure 24) that three genes (gene_18746, gene_7964, and gene_17801) are of far greater importance than the others in terms of their impact on the model output magnitude. While this plots gives us the magnitude of importance, it does not give the direction.

To see both the magnitude and direction of impact we can look at a waterfall plot (figure 25) for a single sample from Class 0. In figure x below, we can see that the base value (E[f(x)] is 2.144. This is the value that would be predicted in the absence of any specific knowledge of the features for the current sample. The updated value for the model after the specifics for the sample are considered, f(x), is -2.834.

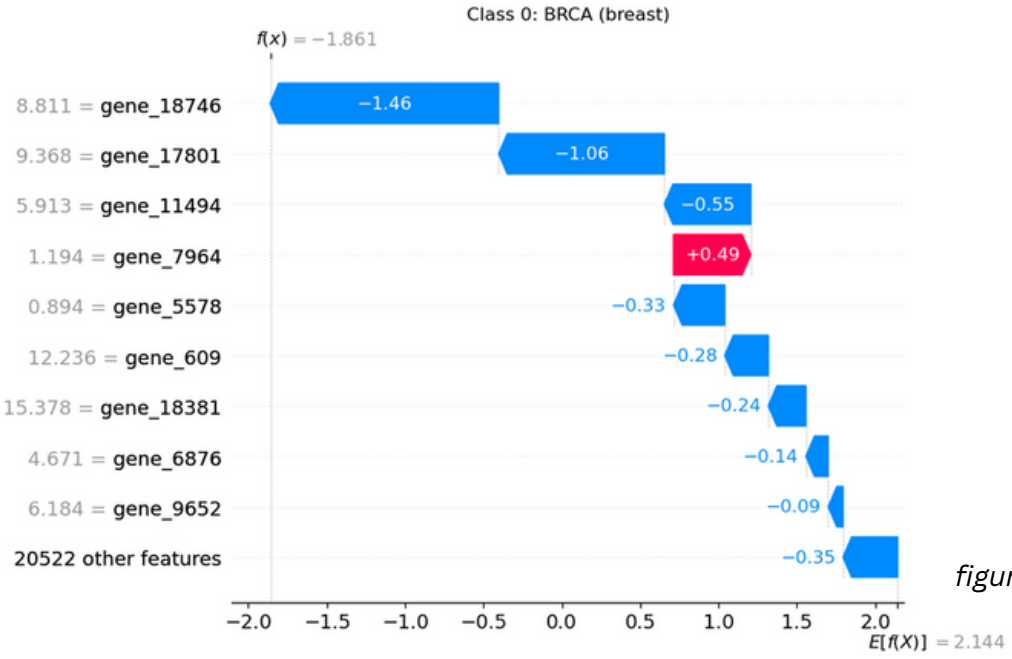


figure 25

The waterfall plot shows that gene_18746 has the greatest impact on this prediction. The gene's value (8.811) causes pushed the score lower by 1.46. The second most important gene in the feature importance summary plot (figure x) was the fourth most influential feature for this particular sample. The value of 1.194 for gene_7964 push the score higher 0.49.

Findings (continued)

The force plot below (figure 26) is similar to the waterfall plot in that it shows both the magnitude and direction of impact for each feature for a single sample. The force plot for this particular feature (not the same feature as was shown in the waterfall plot) show gene_18746 with a value of 14.44 pushing the score higher.

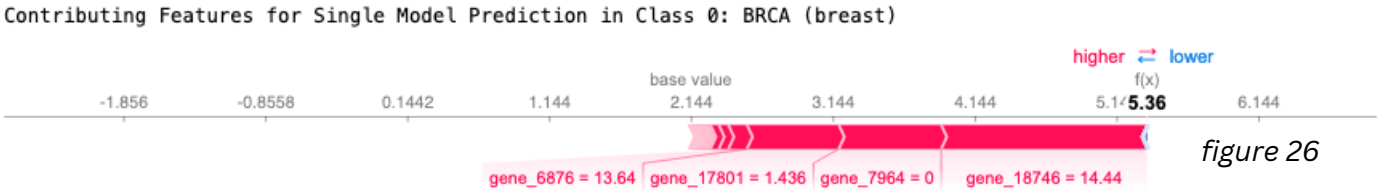


figure 26

Since a lower value of 8.811 pushed the score lower for one sample and the value of 14.44 pushed it higher, might this show a pattern for all samples in this class? We can look at the dot summary plot (figure 27) to gain more insight.

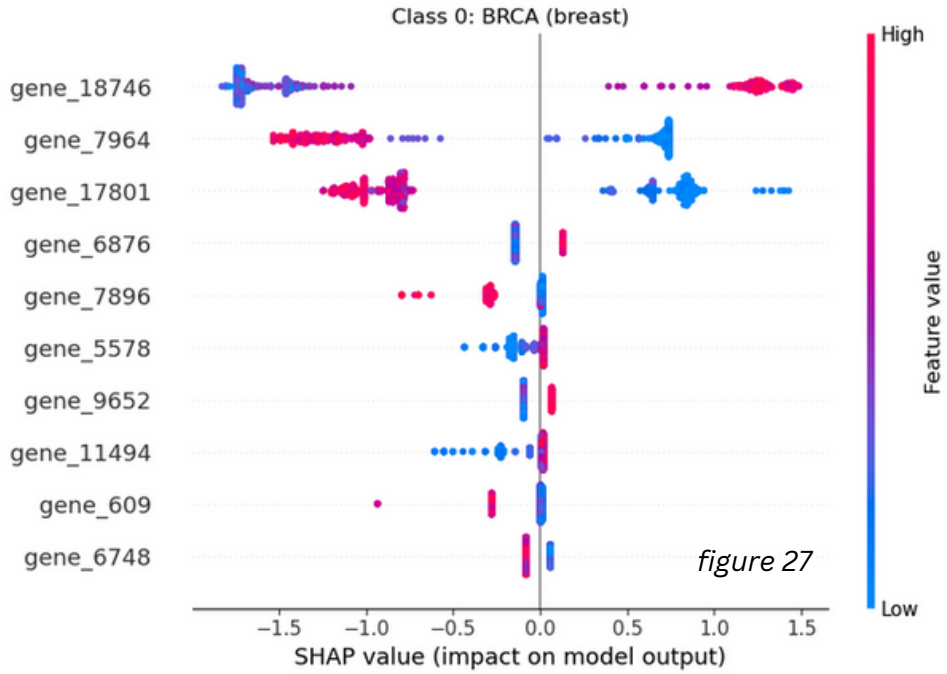


figure 27

The dot version of the summary plot shows that high SHAP values for gene_18746 do tend to map to high feature importance. Lower SHAP values for gene_18746 mostly map to lower values of feature importance, but there are some high values for feature importance mixed in. The results for gene_7964 are the opposite: there seems to be an inverse relationship between SHAP value and feature importance.

We can also take a closer look at the distribution of values for gene_18746 for each class through a swarm plot (figure 28). The values of gene_18746 are generally higher than all of the other classes. The tail, however, crosses the values in all of the other classes, helping us understand the overlap of positive and negative feature values for low SHAP values above. When the value of gene_18746 rises above approximately 11, the model should have no trouble predicting the sample's membership to the BRCA class.

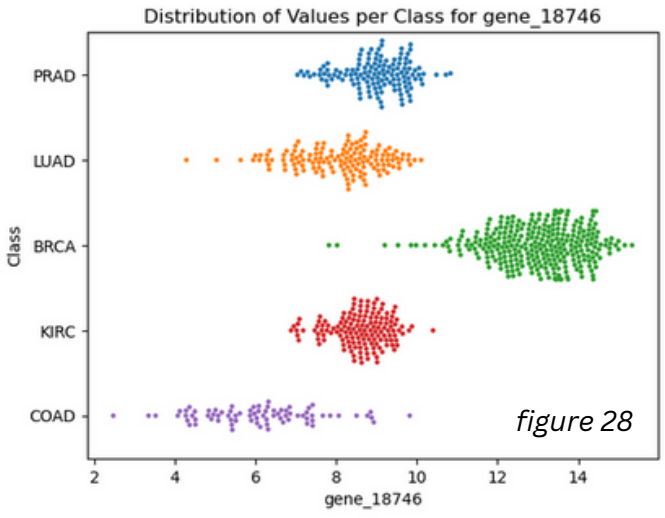


figure 28

Findings (continued)

Class 1: COAD (colon)

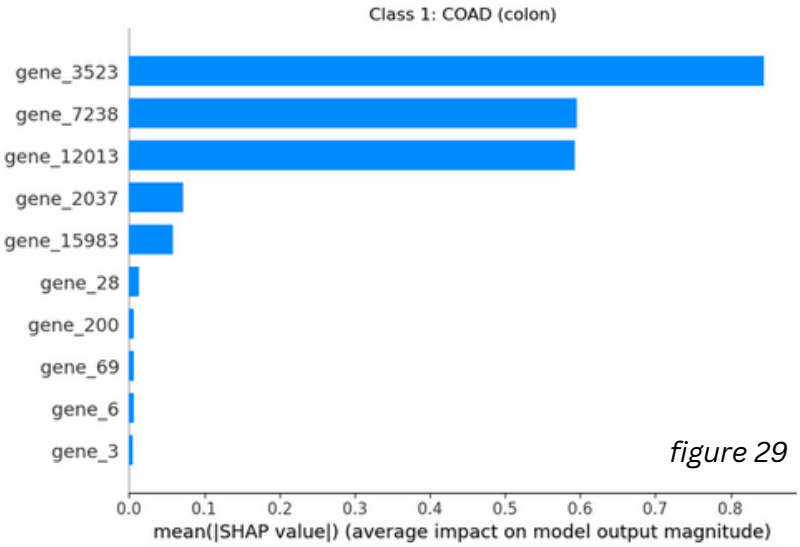


figure 29

As with with breast cancer class, we have three features which stand above the other in terms of the magnitude of their contribution to the prediction of class membership (figure 29). For colon cancer, these are gene_3523, gene_7238, and gene_12013.

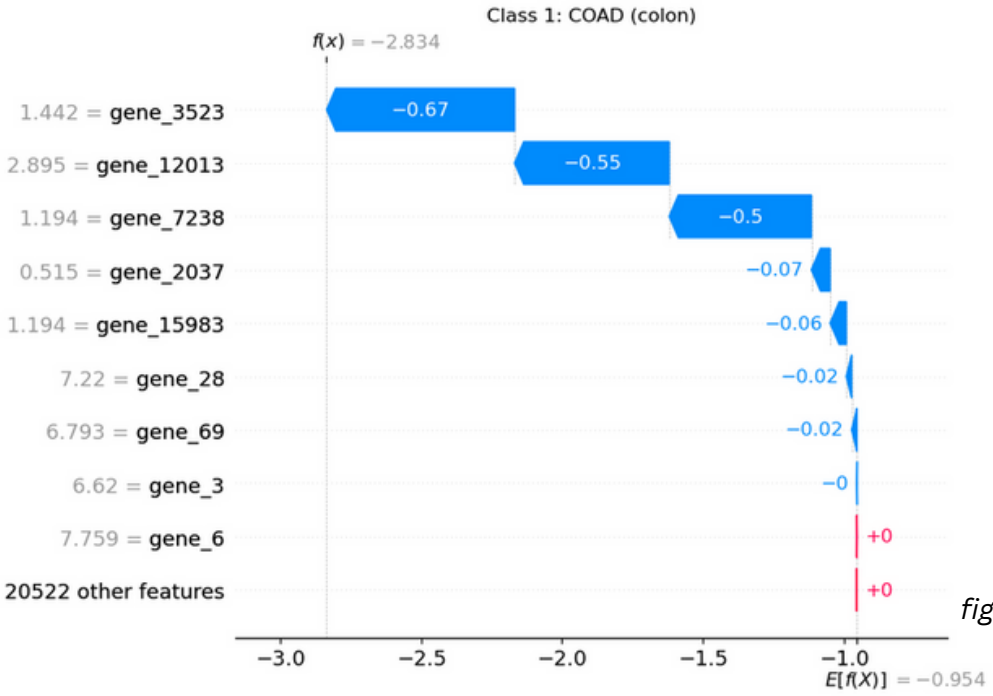


figure 30

In the waterfall plot (figure 30), we see a sample with a value of 1.442 for gene_3523. This value pushes the score lower by 0.67.

Similarly, the sample in the force plot (figure 31) shows gene_3523's value of 0.8354 pushing the score lower.

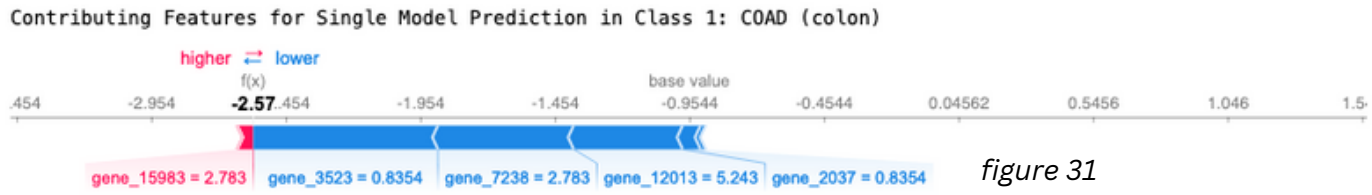


figure 31

Findings (continued)

The dot summary plot (figure 32) shows that high values of gene_3523 map to higher feature importance, although a few occurrences of high importance are also mapped to low SHAP values.

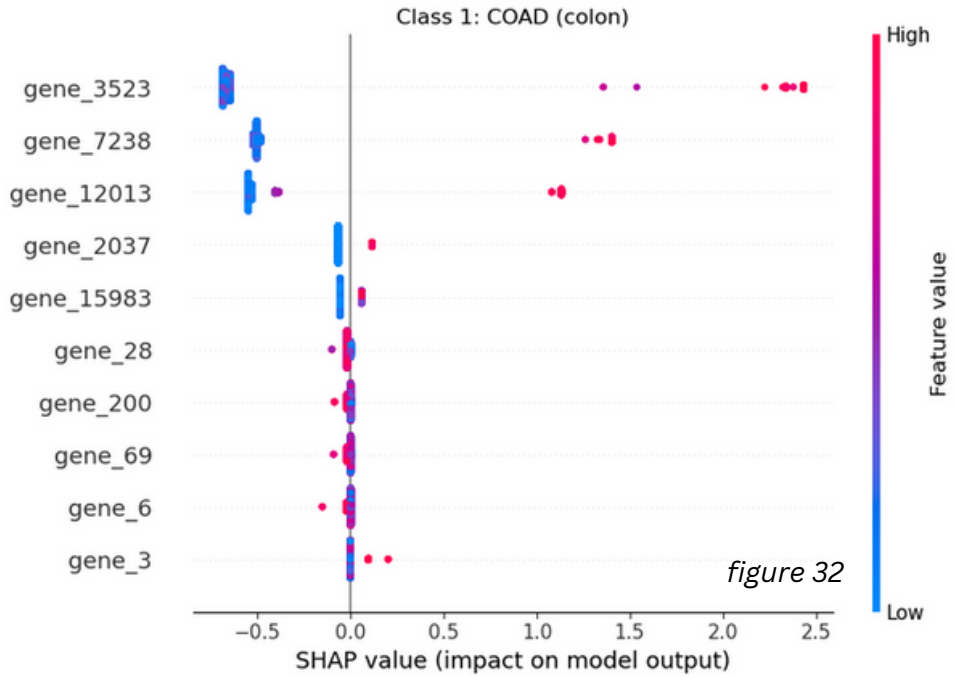


figure 32

The swarm plot (figure 33) provides clarity. When the value of gene_3523 is over 9, the model will have no trouble predicting COAD class membership, but for the relatively few samples with values below that number, this gene alone won't be enough to eliminate class confusion.

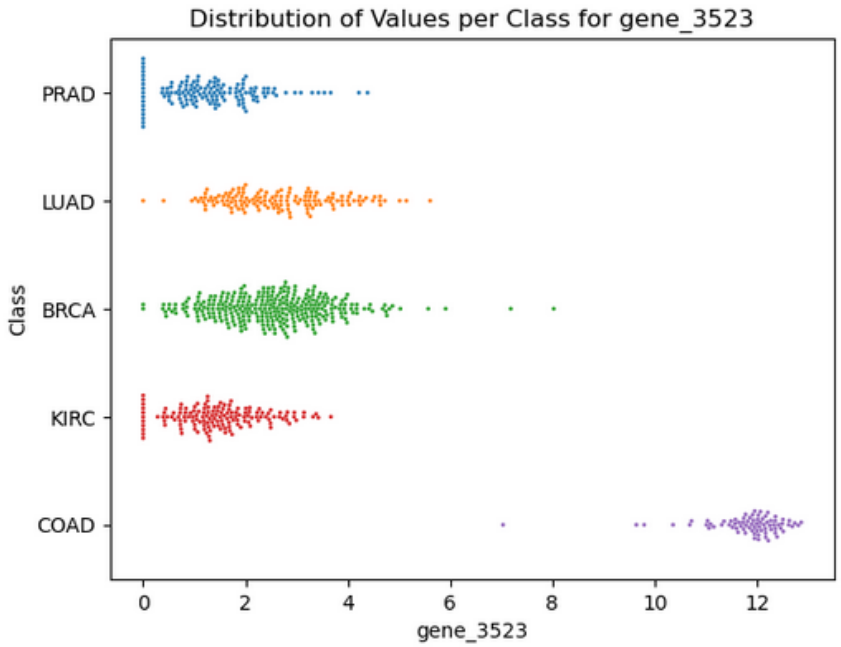


figure 33

Findings (continued)

Class 2: KIRC (kidney)

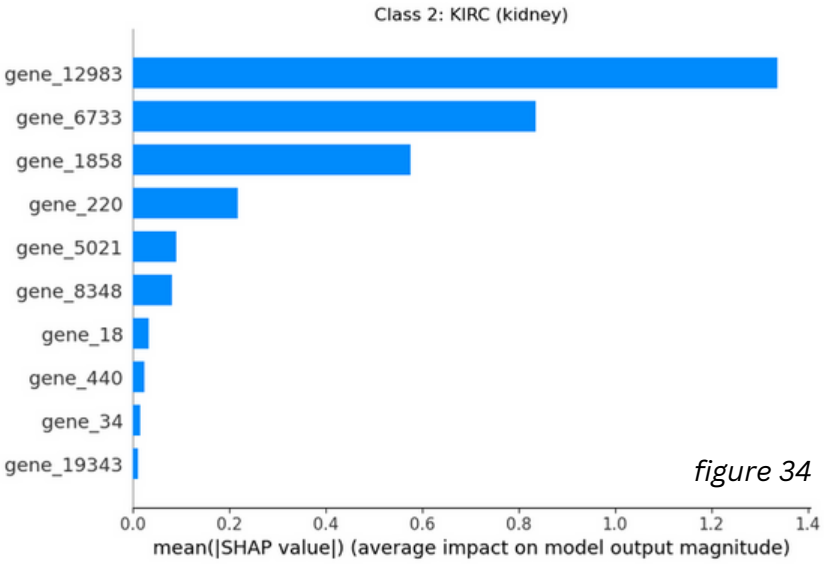


figure 34

While the feature importance plot for Class 2 still shows three features dominating, the amount of difference between the third and the fourth is not as extreme (figure 34).

If we take a look at the first four features in the dot plot (figure 35), we can see that the the high SHAP values very clearly map to high feature importance values. However, the lower SHAP values for gene_12983, while mostly mapped to low SHAP feature importance values, are muddied with a number of higher feature importance values. The lower SHAP values, although not perfect, show a more clearly defined connection to lower feature importance values.

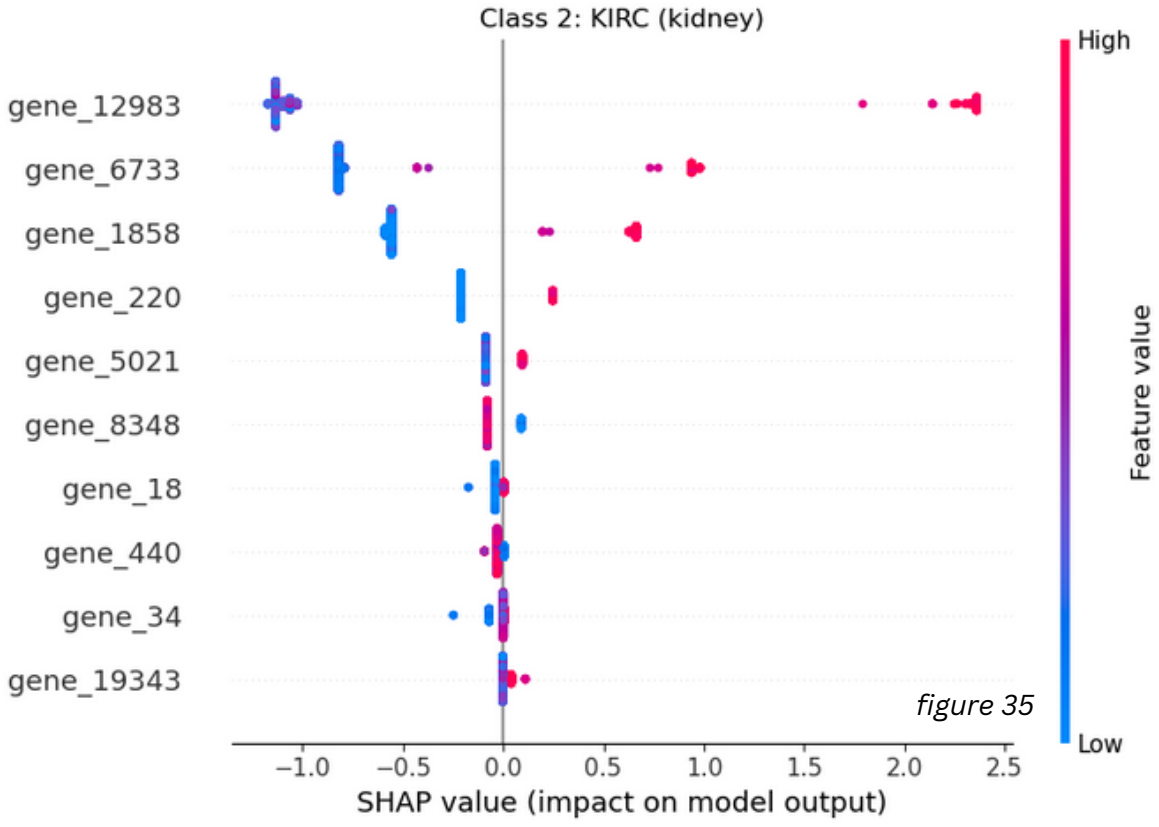


figure 35

Findings (continued)

The swarm plot of gene_12983 (figure 36) shows more clearly the overlap of values for some classes when the value of gene_12983 is below 10.

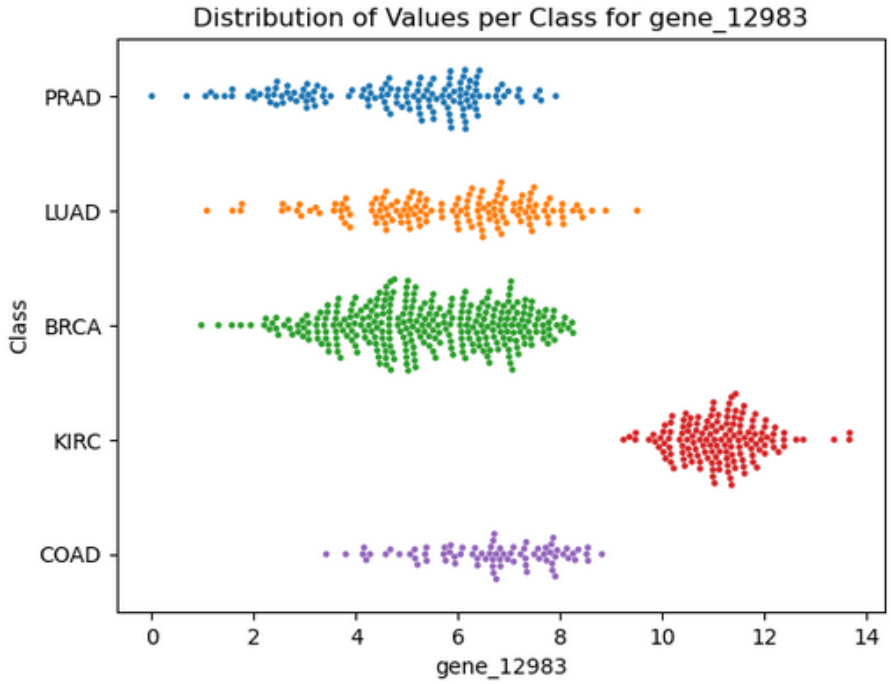


figure 36

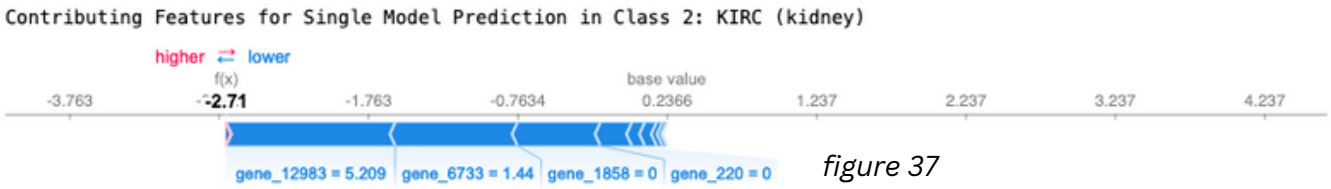


figure 37

We see confirmation of this with the force plot (figure 37) and waterfall plot (figure 38) of two individual samples. Both show low values of gene_12983 pushing the score lower.

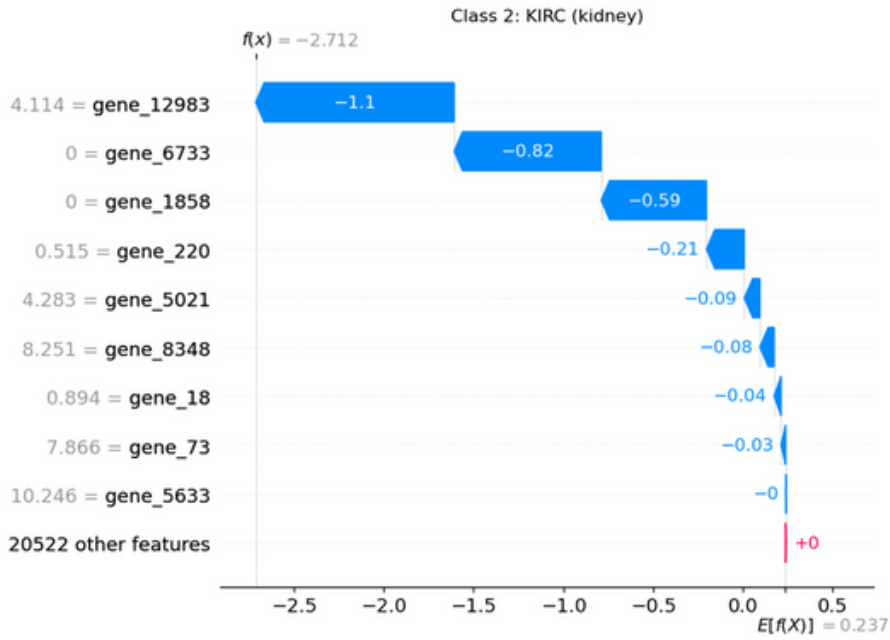


figure 38

Findings (continued)

Class 3: LUAD (lung)

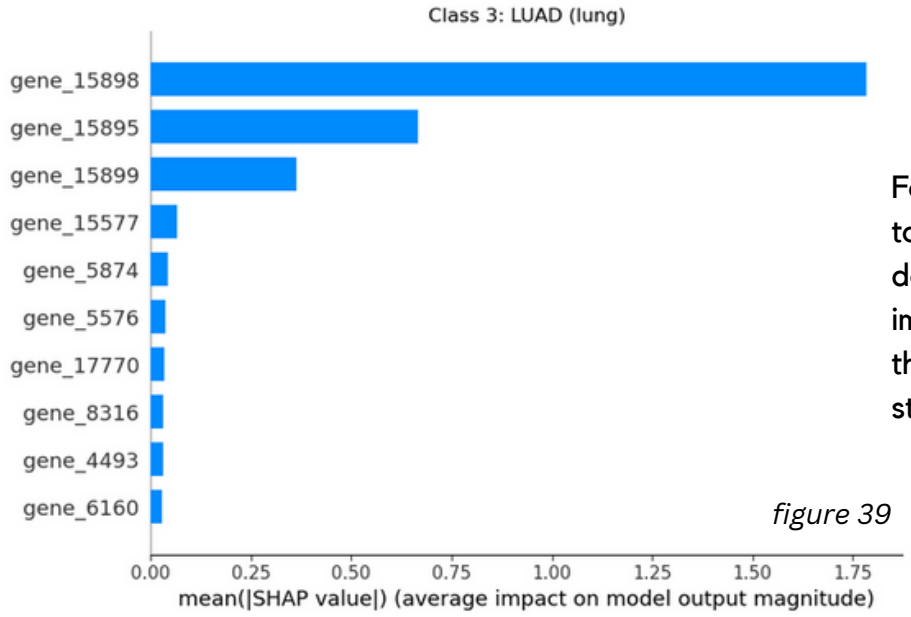


figure 39

For Class 3, you might be inclined to say one feature (gene_15898) dominates in terms of its importance (figure 39). However, I think gene_15895 and gene_15899 still make a good showing.

We can see the top three genes having a notable effect on the model score both in the waterfall plot sample (figure 40) and in the force plot sample (figure 41). Interestingly, though, the value of the gene_15895 and gene_15899 is zero in both cases.

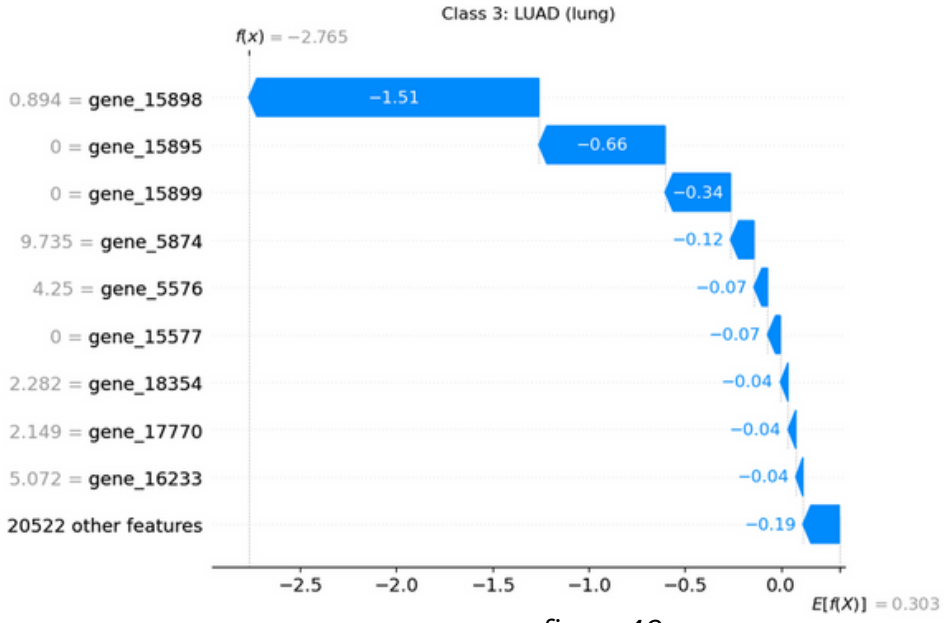


figure 40

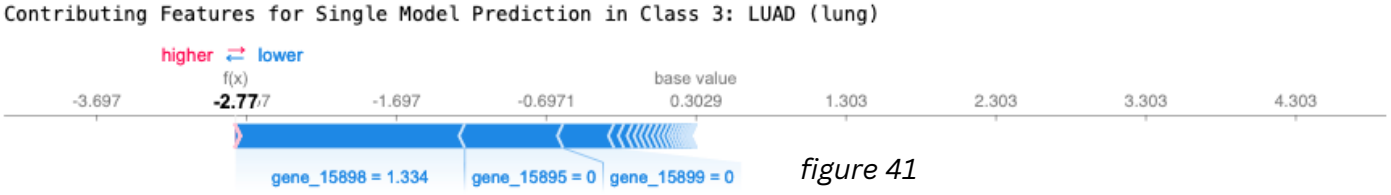


figure 41

Findings (continued)

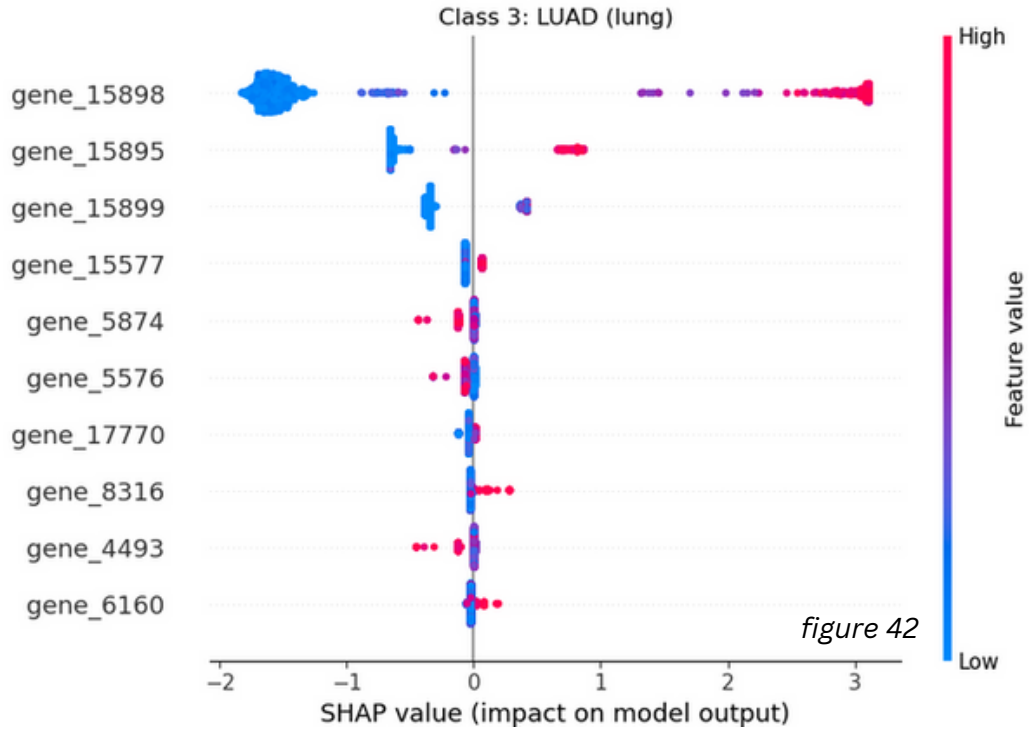


figure 42

The dot summary plot (figure 42) shows a more clear relationship between lower SHAP values and lower feature importance values, but the positive SHAP values between 1 and a little over 2 are quite muddy. Above two and a half or so, the plot shows a clean relationship between the high SHAP values and the high feature importance values.

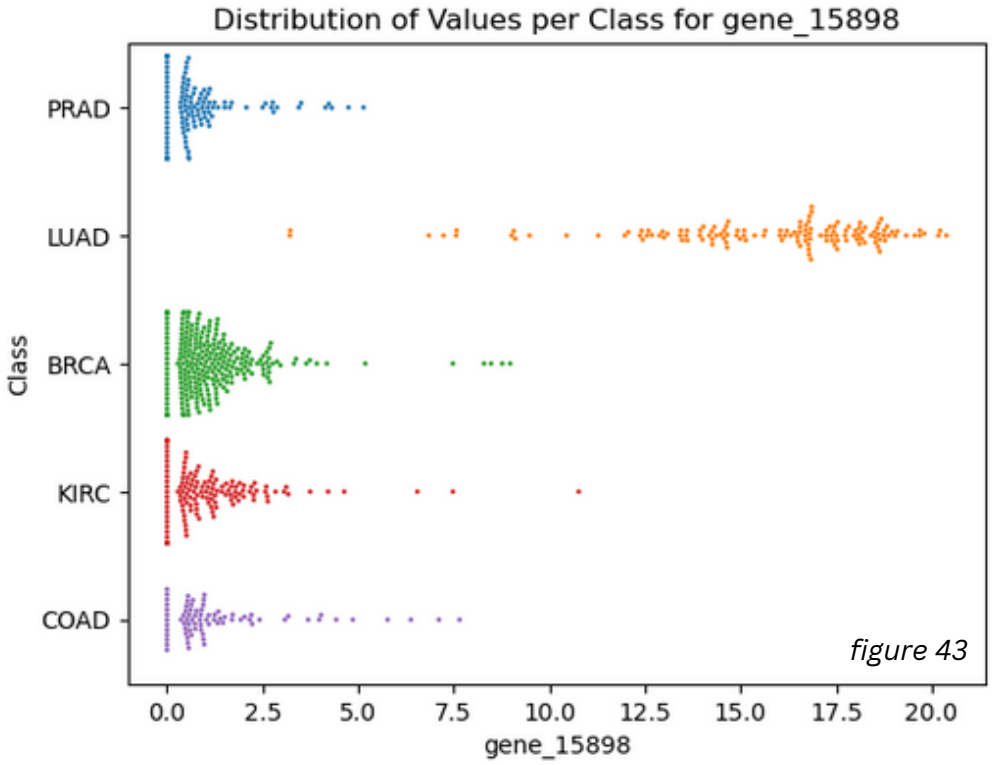


figure 43

The swarm plot for gene_15898 (figure 43) shows a wider range of values for the class we are predicting than we saw for the primary feature from each of the other types of cancer. The tail of lower values is much longer and crosses the high value tails of all the other classes. This makes it more important to have additional features with which to predict lung cancer class membership.

Findings (continued)

Class 4: PRAD (prostate)

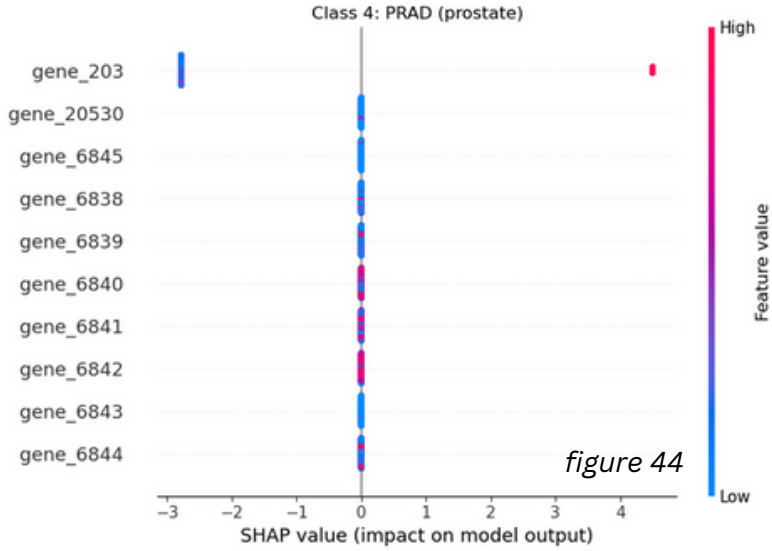


figure 44

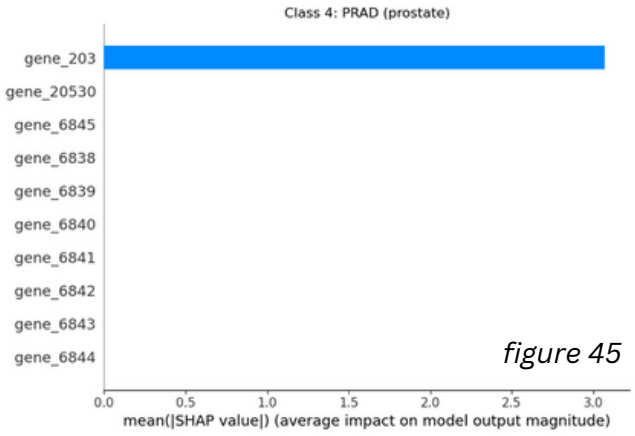


figure 45

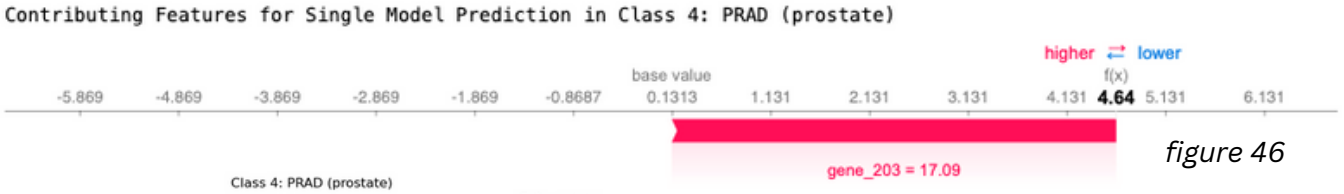


figure 46

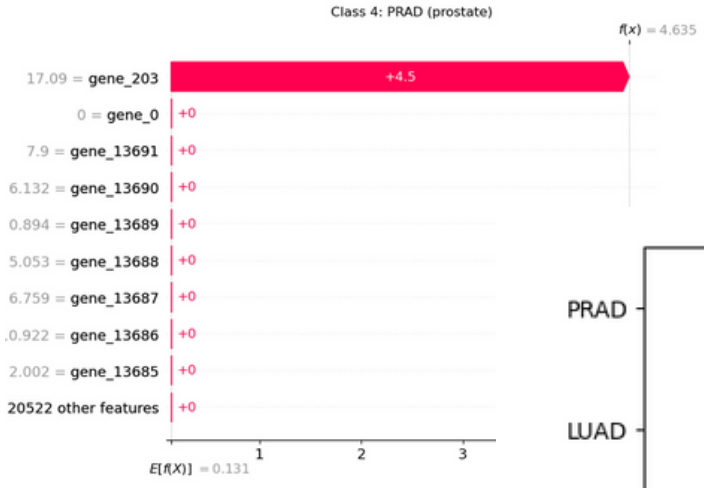


figure 47

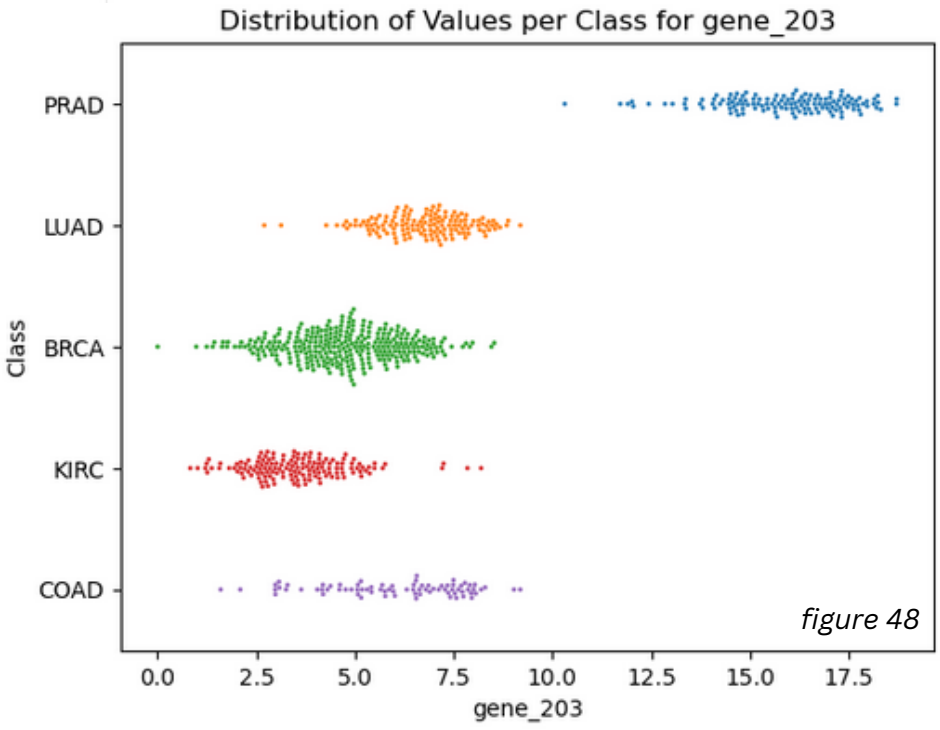


figure 48

Class 4 is unquestionably the easiest to predict. If the value of gene_203 is over 10, you can confidently say it is a member of the prostate cancer class (figures 44-47 and in particular figure 48).

Conclusions & Client Recommendations

All six of the models tried in this project performed very well. They performed so well, in fact, that is might seem you could pick a model out of a hat and have equally good results. However, if a client needs the best results for one class in particular, then one model might be a better choice than another. We can make recommendations to clients based on their needs by referencing the chart below.

	Score	Class 0 BRCA (breast)		Class 1 COAD (colon)		Class 2 KIRC (kidney)		Class 3 LUAD (lung)		Class 4 PRAD (prostate)	
		FP	FN	FP	FN	FP	FN	FP	FN	FP	FN
K Neighbors Classifier	*0.994711*	1	0	0	0	0	0	0	1	0	0
Logistic Regression	*0.994711*	1	0	0	0	0	0	0	1	0	0
Support Vector Classifier	*0.994711*	1	0	0	0	0	0	0	1	0	0
Naive Bayes	*0.984753*	1	0	0	1	0	0	1	1	0	0
XGBoost Classifier	*0.981260*	3	0	0	1	0	0	0	1	0	1
Random Forest Classifier	*0.979361*	2	0	0	1	0	0	1	2	0	0

FP = False Positive (a false positive for class x is one where a true class x sample is predicted to be class y)
FN = False Negative (a false negative for class x is one where a true class y sample is predicted to be class x)

Regardless of your type of cancer a client is focused on, the top three models (N Neighbors Classifier, Logistic Regression, and Support Vector Classifier) will perform well. They are equal in every way - they tied for the highest f1 macro score, and they all have the same number of false positives and false negatives within each class. When it comes to the other three models, though, we start to see some differences.

Breast Cancer:

Breast cancer specialists would do just as well with the Naive Bayes model as they would with the top three. Naive Bayes still has just one false positive and zero false negatives for this class. XGBoost Classifier and Random Forest Classifier show an increase in the number of false positives and so would not be recommended.

Colon Cancer:

Professionals studying colon cancer would do best to stick with one of top three models. Those models make no mistakes in classification for this class but the other three models introduce a false negative.

Kidney Cancer:

All models are equal and excellent for kidney cancer classification. There are no false positives or false negatives for any group.

Lung Cancer:

Clients focusing on lung cancer should choose from the top three models or XGBoost Classifier. Naive Bayes introduces a false positive, and Random Forest Classifier adds both a false positive and an additional false negative.

Prostate Cancer:

Those primarily interested in prostate cancer research would equally as well with all models except XGBoost Classifer. XGBoost introduces on false negative.

Future Work

- Add area under the curve plots for each class.
- Explore the breast cancer sample that is getting misclassified as lung cancer. What is causing this? Are there inaccuracies in the gene values? Is there a different model that would predict this sample accurately without adding any other misclassifications?
- Try the models with new data from alternate source. Would the models still perform as well? Would model strengths and weaknesses become more apparent or less apparent?

Consulted Resources

- The gene expression cancer RNA_Seq data: https://archive-beta.ics.uci.edu/dataset/401/gene+expression+cancer+rna+seq
- A new perspective on Shapley values: https://edden-gerber.github.io/shapley-part-1/
- SHapley Additive exPlanations: https://christophm.github.io/interpretable-ml-book/shap.html
- Measuring differential gene expression with RNA-seq: challenges and strategies for data analysis: https://academic.oup.com/bfg/article/14/2/130/257370?login=false
- Building Machine Learning Clustering Models fro Gene Expression RNA-seq Data: https://ernest-bonat.medium.com/rna-seq-gene-expression-classification-using-machine-learning-algorithms-de862e60bfd0

Packages

- pandas
- matplotlib
- seaborn
- numpy
- sklearn
- xgboost
- SHAP

Appreciation

Special thanks to my mentor, AJ Sanchez, for all of his time, guidance, and encouragement.