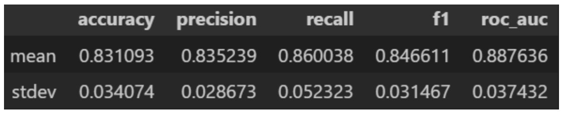
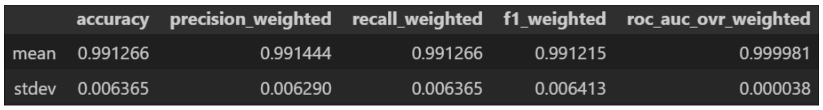
HW2 Writeup

The cancer genomics dataset contains 801 samples, each with 5479 features. This imbalance between the number of samples and the number of features means that this dataset is under the curse of dimensionality, suggesting that overfitting may occur. The tumor classes are slightly imbalanced, with BRCA having 300 samples, while KIRC, LUAD, and PRAD each have around 140 samples, and COAD only has 78. Because of this, I decided to stratify the data when creating train-test splits. I applied standardization to the split sets, dropped duplicates, and converted target labels into numbers so that the GradientBoosting models could process them properly. I used the same preprocessing steps with the heart disease data.

Heart Disease: 

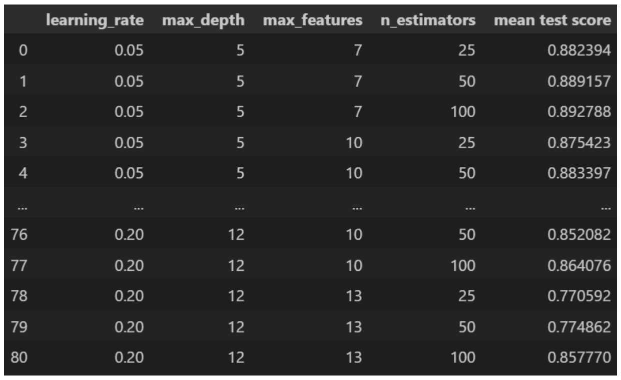
	accuracy	precision	recall	f1	roc_auc
mean	0.831093	0.835239	0.860038	0.846611	0.887636
stdev	0.034074	0.028673	0.052323	0.031467	0.037432

Cancer Genomics: 

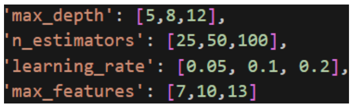
	accuracy	precision_weighted	recall_weighted	f1_weighted	roc_auc_ovr_weighted
mean	0.991266	0.991444	0.991266	0.991215	0.999981
stdev	0.006365	0.006290	0.006365	0.006413	0.000038

The cancer genomics multiclass classification model performed much better than the binary classification model for heart disease. The metrics for the cancer dataset appear concerningly high, but because of the multi-dimensionality of the dataset, near-perfect separation is possible and may be causing those scores. Additionally, there is no data leakage or incorrect splits, suggesting that the high scores are real. The heart disease metrics are very unified overall, showing a reasonably high score for each category. The standard deviation of recall had the highest standard deviation, meaning that across different cross-validation runs, there was higher variance in the model's true positive rate.

For the model hyperparameter tuning, I decided to test different values for max_depth, n_estimators, learning_rate, and max_features. The maximum depth of trees and number of estimators are important hyperparameters for controlling over/underfitting, while learning rate and maximum features evaluated at each split are hyperparameters that relate to efficiency and speed of learning.

Heart Disease Table 

	learning_rate	max_depth	max_features	n_estimators	mean test score
0	0.05	5	7	25	0.882394
1	0.05	5	7	50	0.889157
2	0.05	5	7	100	0.892788
3	0.05	5	10	25	0.875423
4	0.05	5	10	50	0.883397
...	...	...	...	...	...
76	0.20	12	10	50	0.852082
77	0.20	12	10	100	0.864076
78	0.20	12	13	25	0.770592
79	0.20	12	13	50	0.774862
80	0.20	12	13	100	0.857770

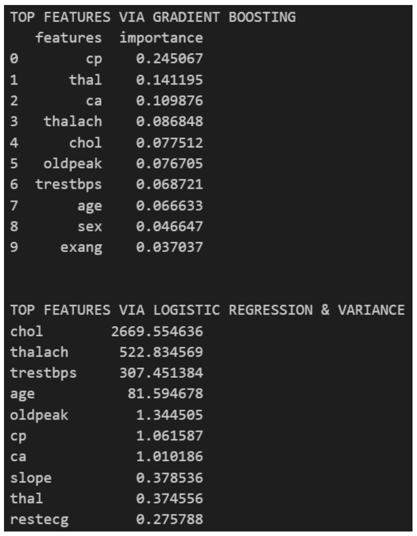


```
'max_depth': [5,8,12],
'n_estimators': [25,50,100],
'learning_rate': [0.05, 0.1, 0.2],
'max_features': [7,10,13]
```

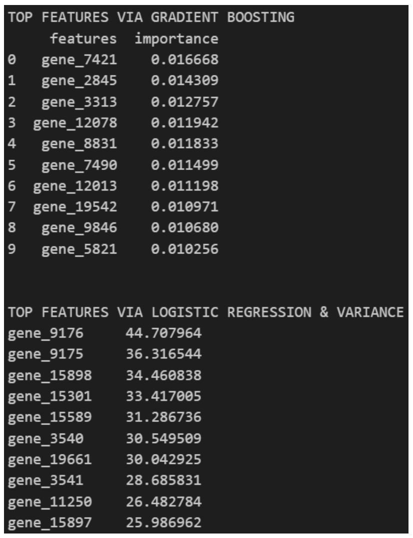
After performing a grid search, the heart disease model was found to perform best with a max depth of 5, 100 estimators, a learning rate of 0.1, and 7 maximum features evaluated at each split. The optimal values of max depth and maximum features are on the lower side of the ranges I chose, while the estimators are on the higher side and the learning rate is in the middle. This indicates that limiting the “scope” of the model but increasing the number of estimators has a positive effect on performance. Ultimately, the best ROC-AUC score achieved by these hyperparameters was 0.898.

The cancer genomics model was found to perform best with a max depth of 8, 50 estimators, a learning rate of 0.05, and 7 maximum features. It is unsurprising that a lower number of maximum features evaluated per split has better performance for a dataset like the cancer genomics, which has many features. However, since the cancer genomics dataset is also highly separable, the effects of hyperparameter tuning are likely not high.

For the heart disease dataset (see 1 below), the gradient boosting model and the HW1 model feature selection method (via variance) disagree on some of the most important features. Gradient boosting put *cp* as the most important by far, while the variance method prioritized *chol*. Otherwise, many of the top features are the same between models, albeit in different orders within the top cohort. For the cancer genomics dataset (2), on the other hand, the two models prioritized quite different gene features. This is not unexpected, because of the sheer number of features in the cancer dataset.

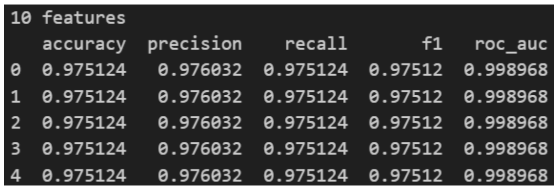
1) 

```
TOP FEATURES VIA GRADIENT BOOSTING
   features  importance
0        cp    0.245067
1      thal    0.141195
2        ca    0.109876
3   thalach    0.086848
4      chol    0.077512
5   oldpeak    0.076705
6  trestbps    0.068721
7       age    0.066633
8       sex    0.046647
9     exang    0.037037


TOP FEATURES VIA LOGISTIC REGRESSION & VARIANCE
chol        2669.554636
thalach      522.834569
trestbps     307.451384
age           81.594678
oldpeak        1.344505
cp             1.061587
ca             1.010186
slope          0.378536
thal           0.374556
restecg        0.275788
```

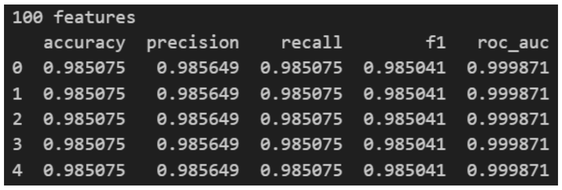
2) 

```
TOP FEATURES VIA GRADIENT BOOSTING
     features  importance
0   gene_7421    0.016668
1   gene_2845    0.014309
2   gene_3313    0.012757
3  gene_12078    0.011942
4   gene_8831    0.011833
5   gene_7490    0.011499
6  gene_12013    0.011198
7  gene_19542    0.010971
8   gene_9846    0.010680
9   gene_5821    0.010256


TOP FEATURES VIA LOGISTIC REGRESSION & VARIANCE
gene_9176     44.707964
gene_9175     36.316544
gene_15898    34.460838
gene_15301    33.417005
gene_15589    31.286736
gene_3540     30.549509
gene_19661    30.042925
gene_3541     28.685831
gene_11250    26.482784
gene_15897    25.986962
```

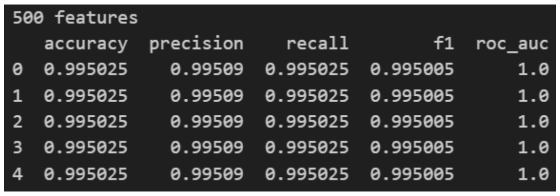
I tested k-feature selection—based on feature importance assigned by the cancer genomics gradient boosted model, but applied to the simple logistic regression model—for k=10, 100, and 500. The full-feature model performed very well as-is, with all metrics having a value of around 0.99, with the exception of ROC-AUC, which was nearly 1. As expected for a highly separable dataset such as this one, model performance and number of features had a positive relationship, which can be seen through the metric values shown below.



```
10 features
   accuracy  precision    recall       f1   roc_auc
0  0.975124   0.976032  0.975124  0.97512  0.998968
1  0.975124   0.976032  0.975124  0.97512  0.998968
2  0.975124   0.976032  0.975124  0.97512  0.998968
3  0.975124   0.976032  0.975124  0.97512  0.998968
4  0.975124   0.976032  0.975124  0.97512  0.998968
```



```
100 features
   accuracy  precision    recall        f1   roc_auc
0  0.985075   0.985649  0.985075  0.985041  0.999871
1  0.985075   0.985649  0.985075  0.985041  0.999871
2  0.985075   0.985649  0.985075  0.985041  0.999871
3  0.985075   0.985649  0.985075  0.985041  0.999871
4  0.985075   0.985649  0.985075  0.985041  0.999871
```



```
500 features
   accuracy  precision    recall        f1   roc_auc
0  0.995025    0.99509  0.995025  0.995005       1.0
1  0.995025    0.99509  0.995025  0.995005       1.0
2  0.995025    0.99509  0.995025  0.995005       1.0
3  0.995025    0.99509  0.995025  0.995005       1.0
4  0.995025    0.99509  0.995025  0.995005       1.0
```

One clear limitation of these models and datasets is that the high separability of the cancer genomics dataset makes it hard to evaluate the model's performance in various situations. Furthermore, because of the separability and the curse of dimensionality, the cancer model may be more prone to overfitting. It is also computationally expensive to train gradient boosting models, especially if hyperparameter tuning is involved, which makes it difficult to test models to the fullest extent.

My multiclass classification model also uses weighted metrics by default, and I do not consider whether macro or micro aggregation might improve performance. For datasets with samples that are more evenly distributed across classes, or for those where rarer classes are not as important, it would be worth testing macro or micro aggregation in addition to weighted.