The lung cancer data used in this Jupyter notebook was published in : Hong, Z.Q. and Yang, J.Y. "Optimal Discriminant Plane for a Small Number of Samples and Design Method of Classifier on the Plane", Pattern Recognition, Vol. 24, No. 4, pp. 317-324, 1991. - Donor: Stefan Aeberhard, stefan at coral.cs.jcu.edu.au - Date : May, 1992

32 lines

57 (1 class attribute, 56 predictive)

attribute 1 is the class label.

- All predictive attributes are nominal, taking on integer values 0-3

Missing Attribute Values: Attributes 5 and 39 (*)

(5 lines have missing data)

9. Class Distribution:

- 3 classes,

1.)	9 observations
2.)	13 "
3.)	10 "

reading the data into:

y = column data 0, the class

x = columns 1 through 56

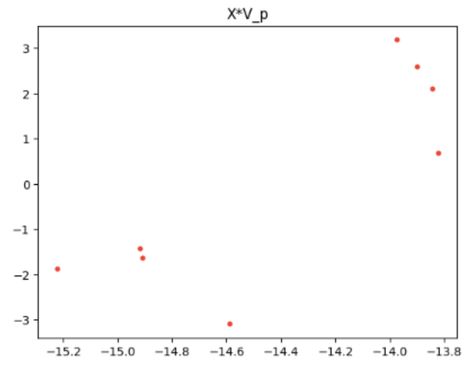
This Jupyter notebook is to explore fitting models with a goal of generalizing well to test data.

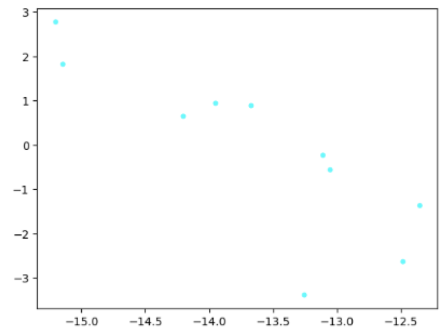
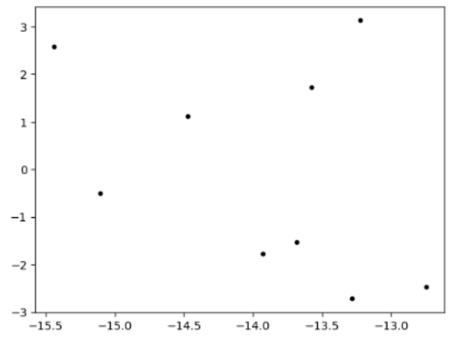
https://github.com/nking/two-point-correlation/blob/master/src/test/python/LungCancer.ipynb

The dataset is small. It is a multi-class classification tabular dataset.
32 (27) rows of 56 features.
There are 3 classes.

A look at SVD based projections of the data separated by labels, reduced to 2 dimensions
shows clear separation and a large first singular value for class 0.

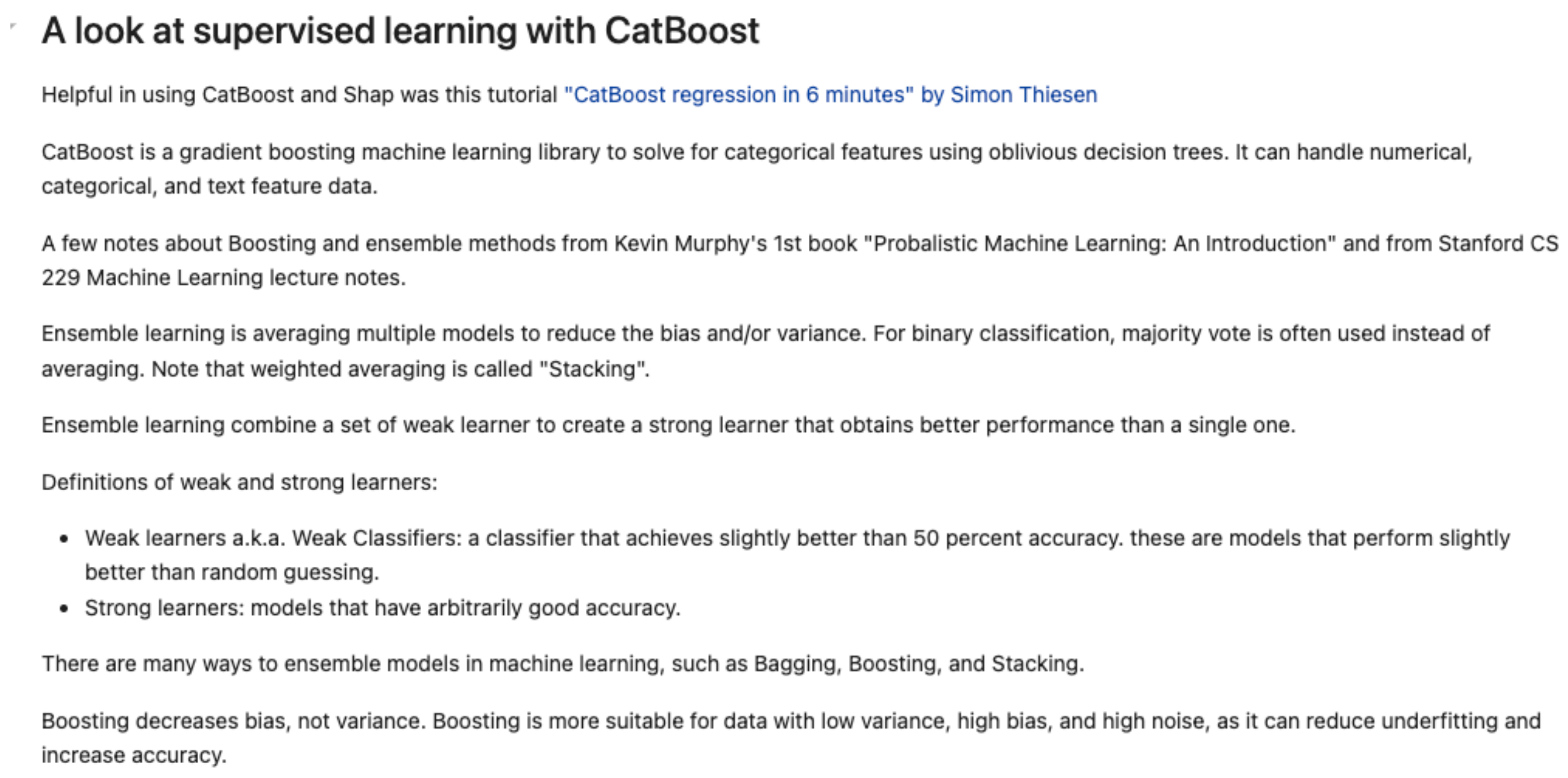
```
class 0 singular values =[40.74947664  6.29577416  4.81186265  4.09467767
3.67063187  3.2651686
  2.31593807  2.10345909]

class 1 singular values =[43.25241679  5.80928818  5.08158973  4.88012914
4.62482867  4.30469475
  3.6134583   3.16877685  2.81159169  2.21803642]

class 2 singular values =[41.89323932  6.3161442   5.36725908  4.76978425
4.10060893  3.49642097
  3.40231263  2.80591209  2.23957204]
```



To use CatBoost, I split the data into train and test. If there were enough data I could split it into train, dev, and test.
For such a small dataset, I used 80% train, 20% test. There are 3 labels, so I chose 80% from each for train, and the rest for test.

After Catboost, I used Shap to make diagrams.



A look at supervised learning with CatBoost

Helpful in using CatBoost and Shap was this tutorial "CatBoost regression in 6 minutes" by Simon Thiesen

CatBoost is a gradient boosting machine learning library to solve for categorical features using oblivious decision trees. It can handle numerical, categorical, and text feature data.

A few notes about Boosting and ensemble methods from Kevin Murphy's 1st book "Probalistic Machine Learning: An Introduction" and from Stanford CS 229 Machine Learning lecture notes.

Ensemble learning is averaging multiple models to reduce the bias and/or variance. For binary classification, majority vote is often used instead of averaging. Note that weighted averaging is called "Stacking".

Ensemble learning combine a set of weak learner to create a strong learner that obtains better performance than a single one.

Definitions of weak and strong learners:

- Weak learners a.k.a. Weak Classifiers: a classifier that achieves slightly better than 50 percent accuracy. these are models that perform slightly better than random guessing.
- Strong learners: models that have arbitrarily good accuracy.

There are many ways to ensemble models in machine learning, such as Bagging, Boosting, and Stacking.

Boosting decreases bias, not variance. Boosting is more suitable for data with low variance, high bias, and high noise, as it can reduce underfitting and increase accuracy.

As an aside, the subnetwork referred to as a "Residual Block" is an example used in Stacking.

Also as an aside, Bagging (Bootstrap Aggregation) is a method of merging the same type of predictions. Bagging decreases variance, not bias, and solves over-fitting issues in a model. The RandomForest method is an example of Bagging.

More regarding CatBoost from dataaspirant and towardsdatascience:

The boosting tree features used in a split are chosen to maximize the split score accuracy which includes a penalty function. The boosting trees are oblivious and balanced trees wherein the same split criterion is used across a level. The feature-split pair chooses a leaf and this is repeated until the leaves reach the tree depth.

And from Bard when asked to summarize leaf growth in CatBoost:

Leaf growth in CatBoost: A summary

Leaf growth in CatBoost refers to the process of building the decision tree at the heart of the model. Here's a breakdown of the key points:

Tree growing methods:

- SymmetricTree: Default approach, builds levels sequentially with fixed splits for all leaves. Fast and often effective, but less flexible than other options.
- Depthwise: Expands by splitting non-terminal leaves with the best loss improvement. More adaptable but not as fast as SymmetricTree.
- Lossguide: Grows leaf by leaf, choosing the non-terminal with the highest gain. Most flexible but not supported for certain analysis methods.

Splitting criteria:

- Minimizes a penalized loss function like cross-entropy or MAE. Penalty terms like L2 regularization prevent overfitting.
- Best split is chosen based on the improvement it brings to the overall loss.

Leaf size limitations:

- min_data_in_leaf: Minimum number of samples allowed in a leaf, controls complexity and prevents noise impact.
- max_leaves: Sets the maximum number of leaves in the tree, another way to control model complexity.

Additional factors:

- Bootstrap aggregation (bagging): Ensembles multiple trees (built on different subsets of data) for better accuracy and robustness.
- Tree depth: Deeper trees can capture more complex relationships but risk overfitting.

Understanding leaf growth is crucial for:

- Tuning CatBoost parameters effectively.
- Interpreting the decision tree's structure and decision rules. Analyzing model complexity and preventing overfitting.

Further resources:

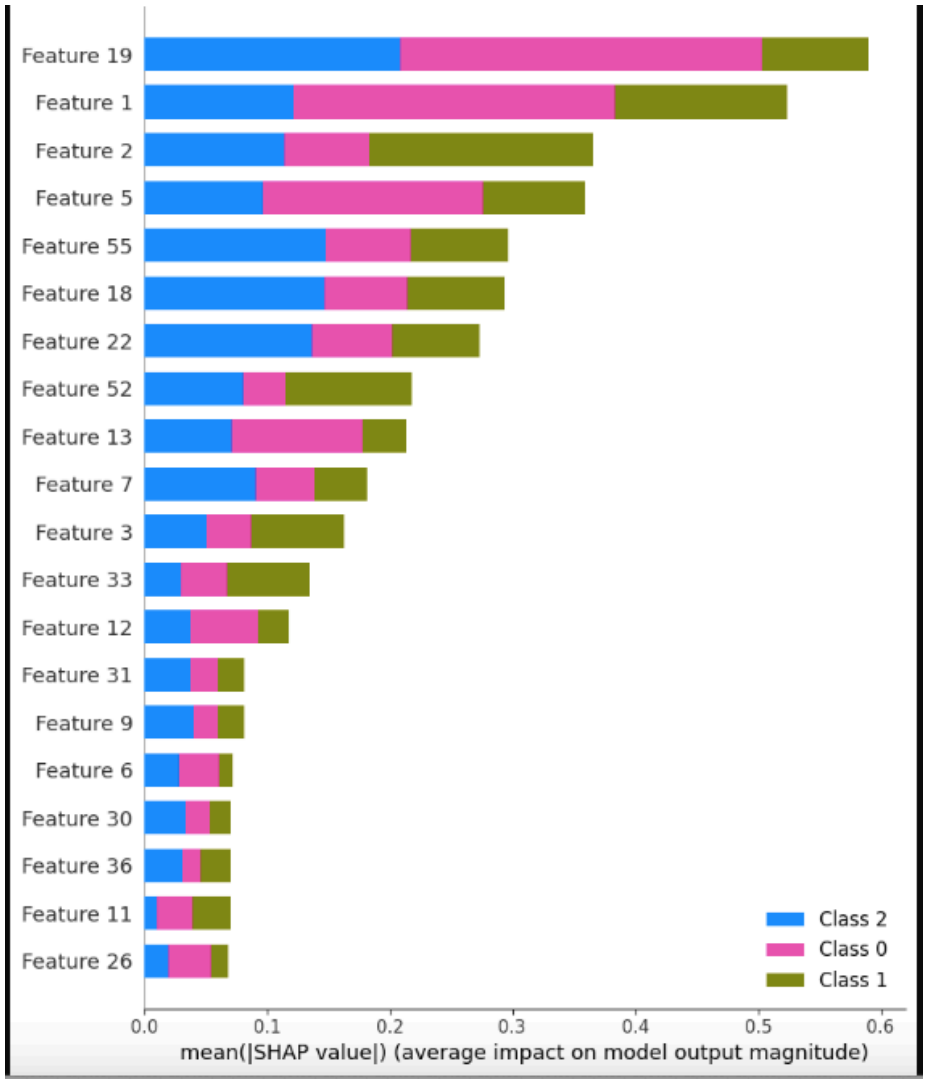
CatBoost documentation on tree growing policies: https://catboost.ai/docs/concepts/parameter-tuning Parameter tuning guide: https://catboost.ai/docs/concepts/parameter-tuning Understanding the CatBoost algorithm: https://medium.com/@harshitaaswani2002/practical-applications-of-catboost-in-data-science-a99f6ff12d00

Using Shap to explore the model results from CatBoost for those with R^2 >= 0.5

Substantial, moderate and weak R^2 scores respectively are 0.75, 0.50, and 0.25

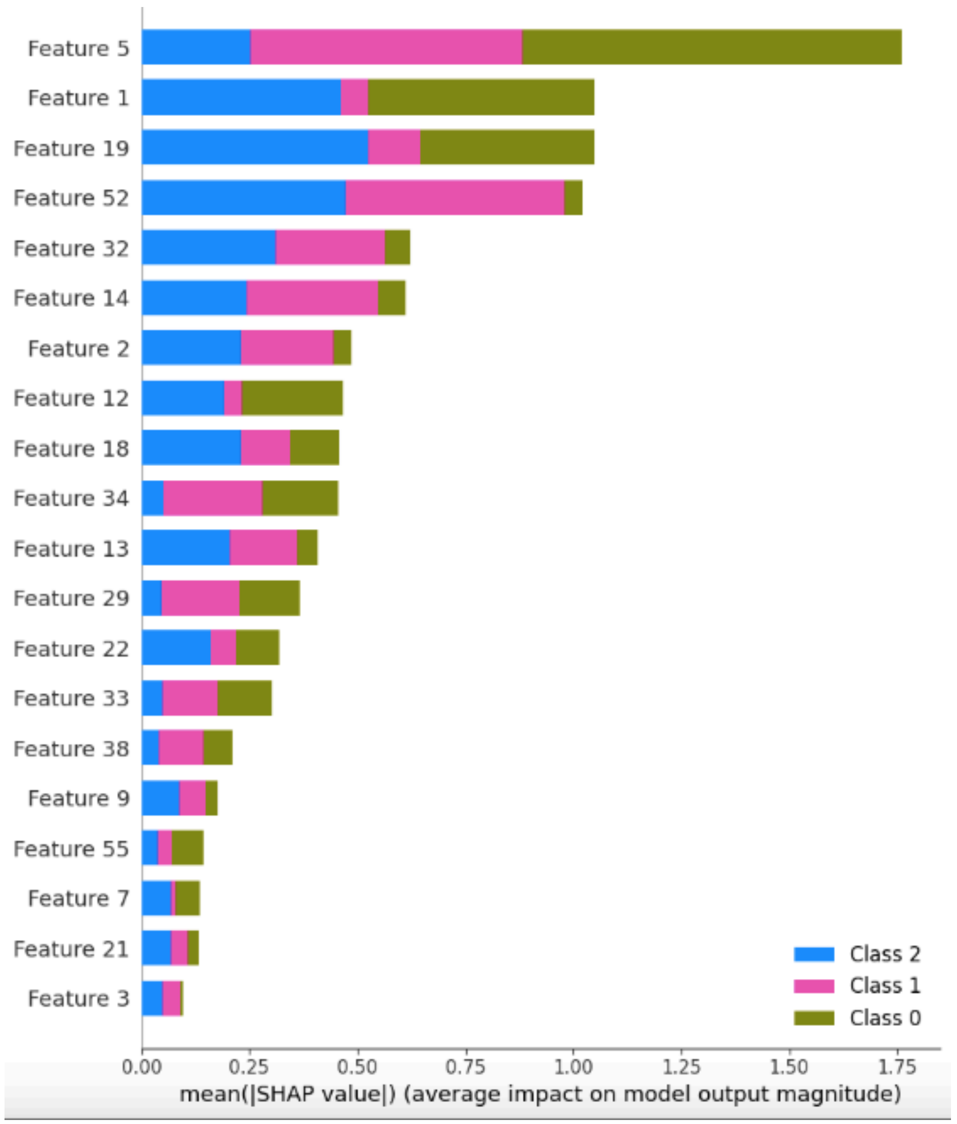
Model 8) 0.50, RMSE: 0.53

params={'loss_function': 'MultiClass', 'verbose': True, 'metric_period': 499, 'classes_count': 3, 'depth': 8, 'l2_leaf_reg': 3, 'learning_rate': 0.03}



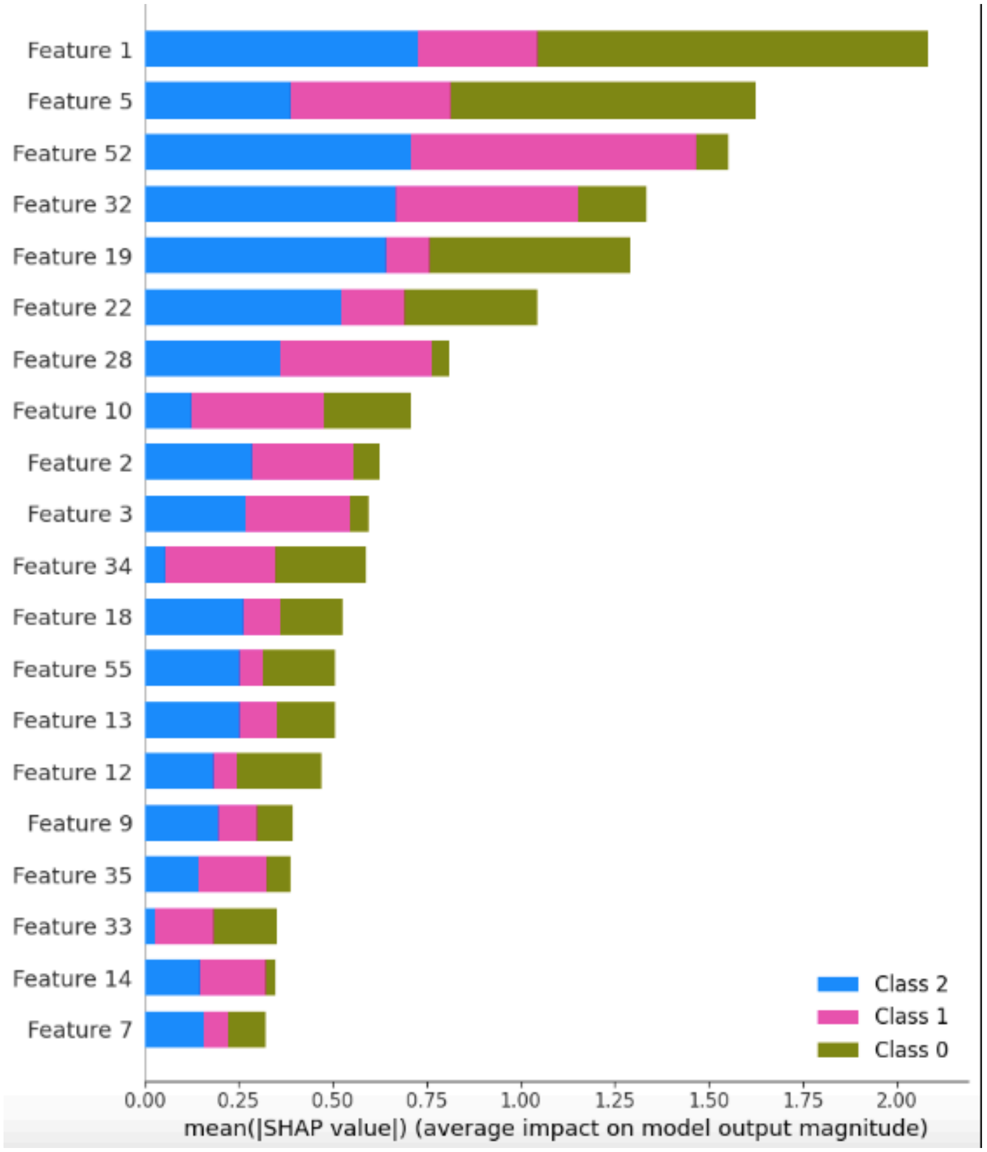
Model 7) R2: 0.50, RMSE: 0.53

params={'loss_function': 'MultiClass', 'verbose': True, 'metric_period': 499, 'classes_count': 3, 'depth': 2, 'l2_leaf_reg': 3, 'learning_rate': 0.1}



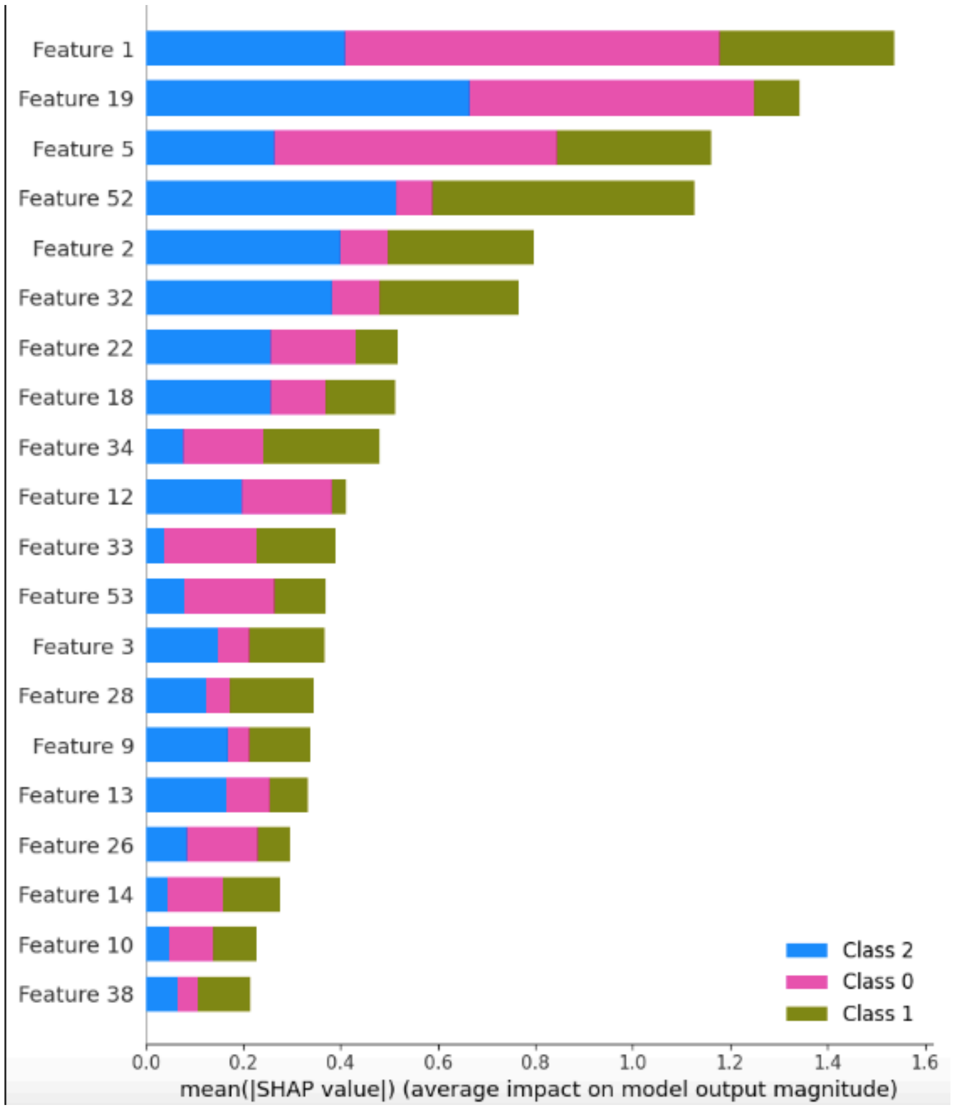
Model 6) R2: 0.50, RMSE: 0.53

params={'loss_function': 'MultiClass', 'verbose': True, 'metric_period': 499, 'classes_count': 3, 'depth': 2, 'learning_rate': 0.1, 'l2_leaf_reg': 0.2}



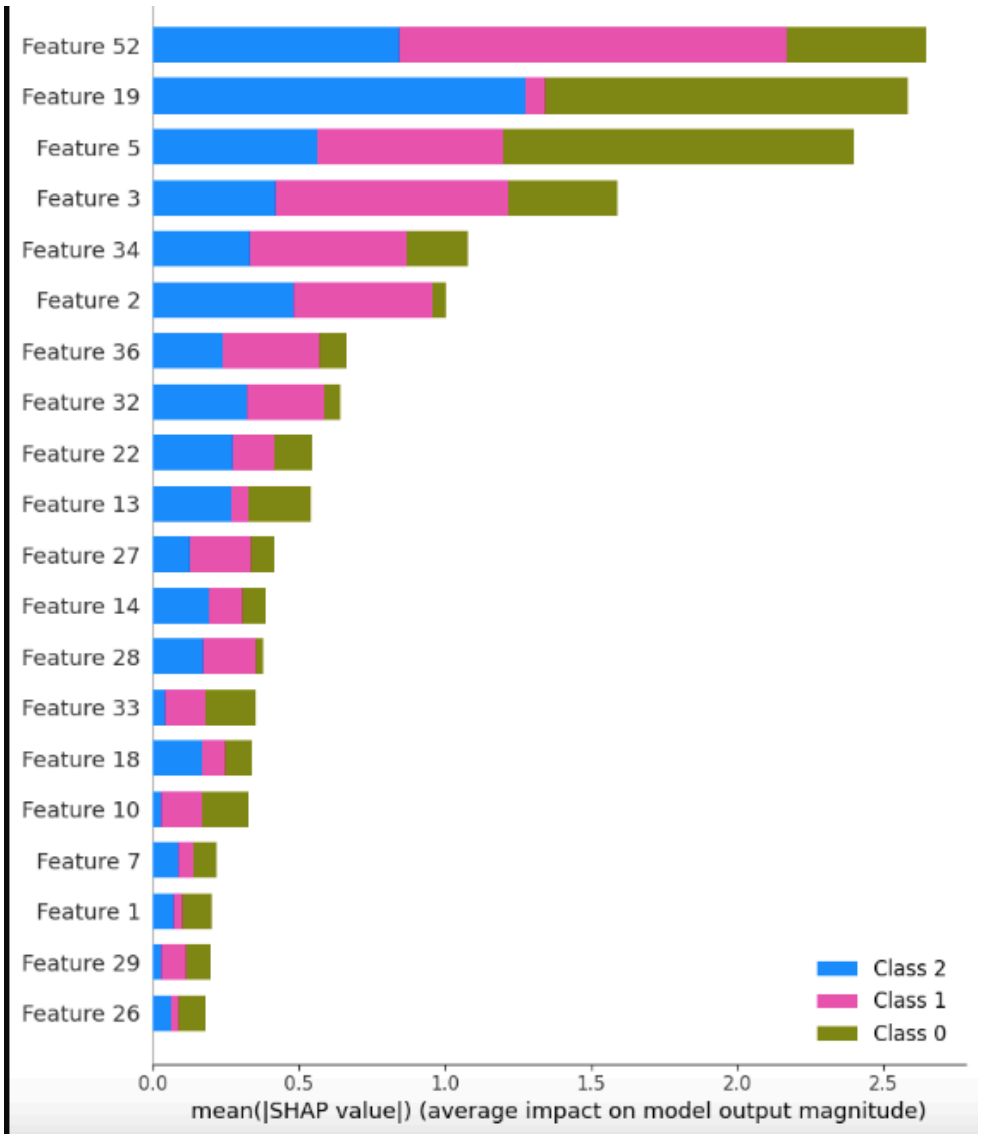
Model 4) R2: 0.50, RMSE: 0.53

params={'loss_function': 'MultiClass', 'verbose': True, 'metric_period': 499, 'classes_count': 3, 'depth': 2, 'learning_rate': 0.03, 'l2_leaf_reg': 0.2}



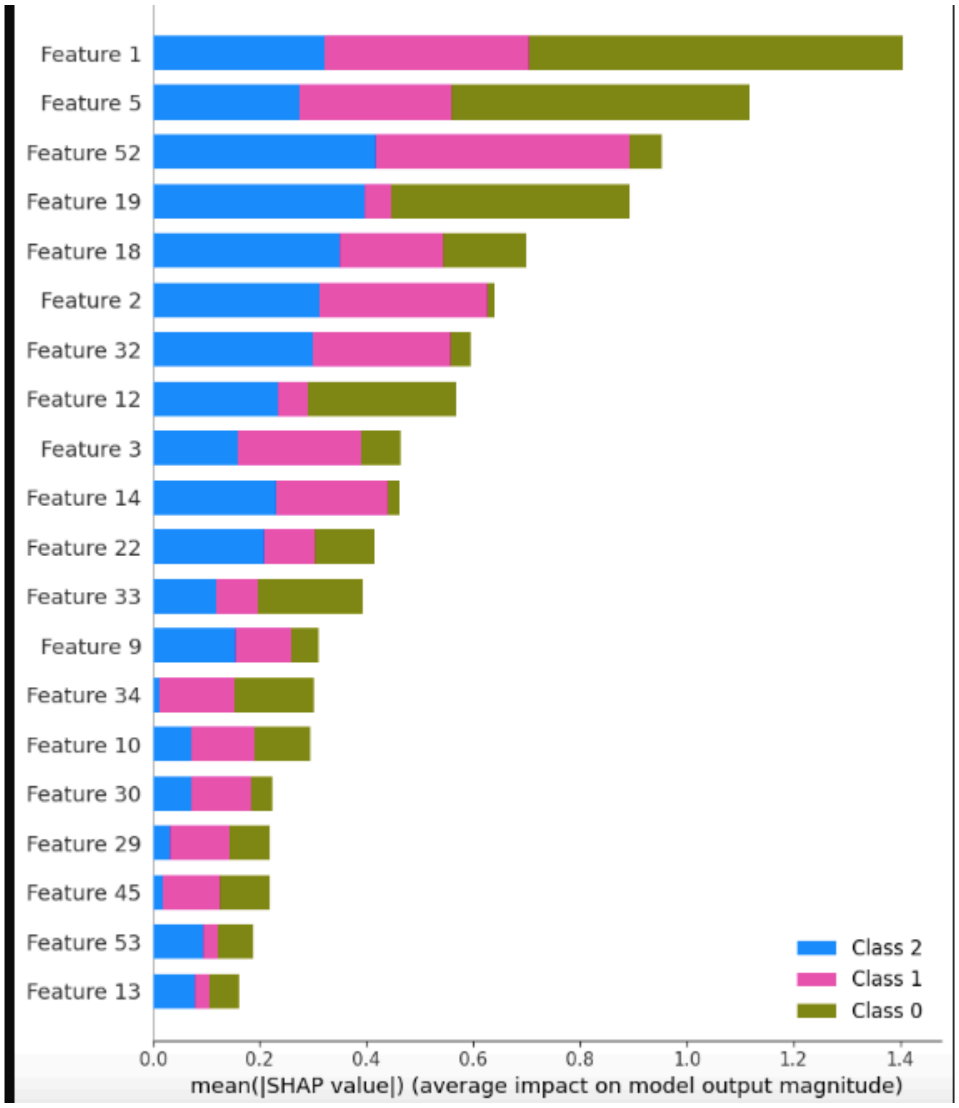
Model 3) R2: 0.50, RMSE: 0.53

params={'loss_function': 'MultiClass', 'verbose': True, 'metric_period': 499, 'classes_count': 3, 'depth': 2, 'learning_rate': 0.1, 'l2_leaf_reg': 0.2}



Model 1) R2: 0.50, RMSE: 0.53

params={'loss_function': 'MultiClass', 'verbose': True, 'metric_period': 499, 'classes_count': 3, 'depth': 2, 'l2_leaf_reg': 3, 'learning_rate': 0.1}


Feature 1
Feature 5
Feature 52
Feature 19
Feature 18
Feature 2
Feature 32
Feature 12
Feature 3
Feature 14
Feature 22
Feature 33
Feature 9
Feature 34
Feature 10
Feature 30
Feature 29
Feature 45
Feature 53
Feature 13
Class 2
Class 1
Class 0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
mean(|SHAP value|) (average impact on model output magnitude)