Joseph Loss (loss2)

IE598 MLF F18

Module 6 Homework (Cross validation)

Part 1: Random test train splits

Sample #	random_state	Accuracy Score
1	1	0.93333
2	2	1.00000
3	3	0.86667
4	4	1.00000
5	5	1.00000
6	6	0.86667
7	7	0.93333
8	8	0.93333
9	9	0.93333
10	10	1.00000

Accuracy Score (Train)	0.985
Mean Score (Test)	0.947
Std. Deviation (Test)	0.050

Part 2: Cross validation

Folds	CV_Accuracy Score
1	0.93333
2	1.00000
3	1.00000
4	0.93333
5	0.93333
6	1.00000
7	1.00000
8	0.91667
9	0.91667
10	0.91667

CV Mean Score (Train)	0.955
CV Std. Deviation (Train)	0.037
CV Accuracy Score (Train)	0.96 +/- 0.037
CV Accuracy (Test)	**0.933**

Part 3: Conclusions

The first method provided the best estimate of how a model will do against unseen data. I believe this is because I tuned the random_state parameter myself, thus having more control over how I wanted the model to behave. However, I'm curious as to why the model returned an accuracy score of 1.0 for several of the random_states and also in the k-folds cross validation. I'd love to discuss this in class and understand the reason for these 100% accuracy scores.

In terms of effectiveness, running the k-folds cross_validation_score would likely be more effective. Using this function in scikit-learn not only leads to more streamlined programming, but also is less verbose in code (which is always better!).

Part 4: Appendix

IE598_F18_HW6