Abalone Regression Report

Raj Patel

I. Introduction to the Dataset

The dataset used for this project contains 4177 datapoints with 8 features and a label each for abalone, a type of sea snail. The underlying supervised learning problem is predicting the rings (age) of an abalone based on sex, length, diameter, height, whole weight, shucked weight, viscera weight, and shell weight. This problem is important to me because it's interesting to see a model built for predicting the age of organic life from its composition.

The metric used to measure performance will be R^2 scoring. Since organic life can pose a lot of variability, the R^2 score would determine how much of the variability is accounted for by the model.

II. Description of the Algorithms

Kernel Ridge Regression with non-linear kernels builds a non-linear model with a slight bias when training to decrease variance when testing. The hyperparameters tested are kernel type, polynomial degree, alpha, and gamma. Increasing gamma would increase the complexity of the hypothesis class; increasing alpha would reduce the complexity of the hypothesis class. Depending on the kernel type and polynomial degree, increasing dimensionality would also increase the complexity of the hypothesis class.

k-Neighbors Regression builds a model by using a distance function to find datapoints closest to each other. The hyperparameters tested are number of neighbors, weight function, and distance metric. All of these hyperparameters don't affect the complexity of the hypothesis class as much as the approach and time complexity of finding the model.

Neural Network with multiple hidden layers builds a model where each layer transforms previous layer values based on its learned weights. The hyperparameters tested are number of hidden layers, activation function, weight optimization solver, alpha, and learning rate. Increasing the number of hidden layers would increase the complexity of the hypothesis class; increasing alpha would reduce the complexity of the hypothesis class. The activation function, weight optimization solver, and learning rate don't affect the complexity of the hypothesis class as much as the approach and time complexity of finding the model.

III. **Tuning Hyperparameters**

Experiment: Tuning hyperparameters will be nested in a 5-fold cross-validation (outer-loop) over the whole dataset. In each fold, the dataset will be split into a training set and testing set beforehand. Hyperparameters will be tuned with grid search 5-fold cross-validation (inner-loop) on the training data for each algorithm. The mean R^2 score of all 5 inner-loop folds will be used to compare hyperparameter performance.

Non-Trivial Justification: The dataset has a non-trivial distribution because a linear regression test results in a score of 0.5348313864079501, which suggests that the data is not strongly linearly correlated.

Results begin on next page.

Table 1: 5-Fold Cross-Validation Hyperparameter Tuning for Kernel Ridge Regression

Outer-Loop Fold #	rank_test_score	mean_test_score	mean_train_score	alpha	degree	gamma	kernel
0	1	0.569732475	0.59699192	0.03		0.5	rbf
0	2	0.569572005	0.601432044	0.03		0.7	rbf
0	3	0.569489076	0.603593853	0.01		0.5	rbf
0	4	0.569390709	0.597764199	0.05		0.7	rbf
0	5	0.569342341	0.597241759	0.01		0.3	rbf
0	6	0.569050216	0.59343011	0.05		0.5	rbf
0	7	0.568930249	0.595134822	0.07		0.7	rbf
0	8	0.568914401	0.601307367	0.05		0.9	rbf
0	9	0.568692892	0.605194713	0.03		0.9	rbf
0	10	0.56863378	0.598542235	0.07		0.9	rbf
1	1	0.579224057	0.596026279	0.01	2	0.7	polynomial
1	2	0.579128087	0.596875821	0.01	2	0.9	polynomial
1	3	0.579051316	0.595126735	0.03	2	1	polynomial
1	4	0.579014459	0.597182056	0.01	2	1	polynomial
1	5	0.578865445	0.594591335	0.03	2	0.9	polynomial
1	6	0.578841423	0.594488851	0.01	2	0.5	polynomial
1	7	0.578566211	0.614167715	0.03		0.9	rbf
1	8	0.578563542	0.616169786	0.03		1	rbf
1	9	0.578426064	0.593644232	0.05	2	1	polynomial
1	10	0.578385176	0.61167423	0.05		1	rbf
2	1	0.575937898	0.609268596	0.01		0.5	rbf
2	2	0.575627871	0.606842918	0.03		0.7	rbf
2	3	0.575408768	0.601554642	0.01		0.3	rbf
2	4	0.575347481	0.601468722	0.03		0.5	rbf
2	5	0.574887501	0.602546582	0.05		0.7	rbf
2	6	0.574581657	0.61109836	0.03		0.9	rbf
2	7	0.57453693	0.606715948	0.05		0.9	rbf
2	8	0.574241228	0.614843434	0.01		0.7	rbf
2	9	0.57405621	0.597187777	0.05		0.5	rbf
2	10	0.574000422	0.608544125	0.05		1	rbf
3	1	0.584409476	0.619536429	0.03		1	rbf
3	2	0.584261601	0.617339804	0.03		0.9	rbf
3	3	0.584099034	0.626539143	0.01		0.9	rbf
3	4	0.584068933	0.621237337	0.01		0.7	rbf
3	5	0.583753732	0.628806089	0.01		1	rbf
3	6	0.583272799	0.614635818	0.05		1	rbf
3	7	0.583241098	0.612309012	0.03		0.7	rbf
3	8	0.583069353	0.61460592	0.01		0.5	rbf

3	9	0.582933078	0.612490565	0.05		0.9	rbf
3	10	0.581998236	0.611113951	0.07		1	rbf
4	1	0.56566813	0.595568142	0.03		0.5	rbf
4	2	0.565523595	0.591824197	0.05		0.5	rbf
4	3	0.565481887	0.59637467	0.05		0.7	rbf
4	4	0.565392256	0.600329018	0.03		0.7	rbf
4	5	0.565325915	0.593594468	0.07		0.7	rbf
4	6	0.565236424	0.58916112	0.07		0.5	rbf
4	7	0.565082426	0.591410718	0.09		0.7	rbf
4	8	0.564960006	0.595856998	0.01		0.3	rbf
4	9	0.564943596	0.590464735	0.1		0.7	rbf
4	10	0.564925867	0.600160431	0.05		0.9	rbf

Table 1 shows the top 5 inner-loop mean R^2 score out of all different hyperparameter combinations used when tuning kernel ridge regression for each outer-loop fold. To keep the report brief, the rest of the results are available in data/report_analysis/krr_i.csv, where i refers to an outer-loop fold number.

Kernel Ridge Regression Fold 0: The final hyperparameters chosen were alpha=0.03, gamma=0.5, and kernel=rbf; the mean validation score of these chosen values was 0.5697324749463106 and mean training score was 0.59699192. Total computation time necessary to run the inner-loop experiment was 432.1549704074862 seconds.

Kernel Ridge Regression Fold 1: The final hyperparameters chosen were alpha=0.01, gamma=0.7, degree=2, and kernel=polynomial; the mean validation score of these chosen values was 0.5792240565175798 and mean training score was 0.596026279. Total computation time necessary to run the inner-loop experiment was 424.82907581329357 seconds.

Kernel Ridge Regression Fold 2: The final hyperparameters chosen were alpha=0.01, gamma=0.5, and kernel=rbf; the mean validation score of these chosen values was 0.575937898351506 and mean training score was 0.609268596. Total computation time necessary to run the inner-loop experiment was 425.15480351448076 seconds.

Kernel Ridge Regression Fold 3: The final hyperparameters chosen were alpha=0.03, gamma=1, and kernel=rbf; the mean validation score of these chosen values was 0.5844094763769536 and mean training score was 0.619536429. Total computation time necessary to run the inner-loop experiment was 463.3424472808838 seconds.

Kernel Ridge Regression Fold 4: The final hyperparameters chosen were alpha=0.03, gamma=0.5, and kernel=rbf; the mean validation score of these chosen values was 0.5656681298629453 and mean training score was 0.595568142. Total computation time necessary to run the inner-loop experiment was 493.66556119918806 seconds.

Table 2: 5-Fold Cross-Validation Hyperparameter Tuning for k-Neighbors Regression

Outer-Loop Fold #	rank_test_score	mean_test_score	mean_train_score	metric	n_neighbors	weights
0	1	0.531483302	1	euclidean	15	distance
0	1	0.531483302	1	minkowski	15	distance
0	3	0.531402568	1	euclidean	17	distance
0	3	0.531402568	1	minkowski	17	distance
0	5	0.531213398	1	manhattan	13	distance
0	6	0.531000467	1	manhattan	15	distance
0	7	0.530992231	1	euclidean	19	distance
0	7	0.530992231	1	minkowski	19	distance
0	9	0.529894817	1	euclidean	13	distance
0	9	0.529894817	1	minkowski	13	distance
1	1	0.543371453	1	manhattan	15	distance
1	2	0.54326698	1	euclidean	17	distance
1	2	0.54326698	1	minkowski	17	distance
1	4	0.541584759	1	euclidean	15	distance
1	4	0.541584759	1	minkowski	15	distance
1	6	0.541233716	1	euclidean	19	distance
1	6	0.541233716	1	minkowski	19	distance
1	8	0.541167735	0.596255798	manhattan	15	uniform
1	9	0.540839799	1	manhattan	13	distance
1	10	0.540493591	1	manhattan	17	distance
2	1	0.527466026	1	euclidean	13	distance
2	1	0.527466026	1	minkowski	13	distance
2	3	0.527011613	1	euclidean	15	distance
2	3	0.527011613	1	minkowski	15	distance
2	5	0.526651196	1	euclidean	11	distance
2	5	0.526651196	1	minkowski	11	distance
2	7	0.526226251	1	euclidean	17	distance
2	7	0.526226251	1	minkowski	17	distance
2	9	0.526223682	1	manhattan	13	distance
2	10	0.525569657	1	manhattan	15	distance
3	1	0.541393102	1	euclidean	15	distance
3	1	0.541393102	1	minkowski	15	distance
3	3	0.539908956	1	euclidean	17	distance
3	3	0.539908956	1	minkowski	17	distance
3	5	0.539049563	1	euclidean	19	distance
3	5	0.539049563	1	minkowski	19	distance
3	7	0.538965804	1	euclidean	13	distance
3	7	0.538965804	1	minkowski	13	distance

3	9	0.538119095	1	manhattan	17	distance
3	10	0.537941304	0.59740856	euclidean	15	uniform
3	10	0.537941304	0.59740856	minkowski	15	uniform
4	1	0.533628013	1	euclidean	19	distance
4	1	0.533628013	1	minkowski	19	distance
4	3	0.532210024	1	euclidean	17	distance
4	3	0.532210024	1	minkowski	17	distance
4	5	0.529436606	1	chebyshev	19	distance
4	6	0.52922255	1	manhattan	19	distance
4	7	0.529154104	1	euclidean	15	distance
4	7	0.529154104	1	minkowski	15	distance
4	9	0.529107659	0.574993024	euclidean	19	uniform
4	9	0.529107659	0.574993024	minkowski	19	uniform

Table 2 shows the top 5 inner-loop mean R^2 score out of all different hyperparameter combinations used when tuning k-Neighbors regression for each outer-loop fold. To keep the report brief, the rest of the results are available in data/report_analysis/knr_i.csv, where i refers to an outer-loop fold number.

k-Neighbors Regression Fold 0: The final hyperparameters chosen were n_neighbors=15, metric=euclidean, and weights=distance; the mean validation score of these chosen values was 0.5314833016301584 and mean training score was 1. Total computation time necessary to run the inner-loop experiment was 3.9277629852294913 seconds.

k-Neighbors Regression Fold 1: The final hyperparameters chosen were n_neighbors=15, metric=manhattan, and weights=distance; the mean validation score of these chosen values was 0.5433714530899346 and mean training score was 1. Total computation time necessary to run the inner-loop experiment was 4.363337993621826 seconds.

k-Neighbors Regression Fold 2: The final hyperparameters chosen were n_neighbors=13, metric=euclidean, and weights=distance; the mean validation score of these chosen values was 0.5274660258709062 and mean training score was 1. Total computation time necessary to run the inner-loop experiment was 4.379255771636963 seconds.

k-Neighbors Regression Fold 3: The final hyperparameters chosen were n_neighbors=15, metric=euclidean, and weights=distance; the mean validation score of these chosen values was 0.5413931023831708 and mean training score was 1. Total computation time necessary to run the inner-loop experiment was 4.372203350067139 seconds.

k-Neighbors Regression Fold 4: The final hyperparameters chosen were n_neighbors=19, metric=euclidean, and weights=distance; the mean validation score of these chosen values was 0.5336280127137727 and mean training score was 1. Total computation time necessary to run the inner-loop experiment was 4.985399007797241 seconds.

Table 3: 5-Fold Cross-Validation Hyperparameter Tuning for Neural Network

Outer-Loop Fold #	rank_test_score	mean_test_score	mean_train_score	activation	alpha	hidden_layer_sizes	learning_rate_init	solver
0	1	0.586106982	0.599733948	tanh	1.00E-05	(3,)	0.01	lbfgs
0	2	0.585910826	0.600246309	logistic	0.0001	(3,)	0.1	lbfgs
0	3	0.585902886	0.595819003	logistic	1.00E-05	(2,)	0.001	lbfgs
0	4	0.585715949	0.595823991	logistic	0.0001	(2,)	0.001	lbfgs
0	5	0.585558153	0.601540744	logistic	0.01	(3,)	0.01	lbfgs
0	6	0.585545785	0.595774018	logistic	0.001	(2,)	0.1	lbfgs
0	7	0.585506454	0.596049883	tanh	1.00E-05	(2,)	0.01	lbfgs
0	8	0.585503936	0.595631265	logistic	0.01	(2,)	0.001	lbfgs
0	9	0.585432544	0.595837886	logistic	0.0001	(2,)	0.01	lbfgs
0	10	0.585421376	0.595987864	tanh	1.00E-05	(2,)	0.1	lbfgs
1	1	0.593367601	0.604681557	tanh	0.1	(2,)	0.01	lbfgs
1	2	0.593303522	0.606359702	logistic	0.1	(3,)	0.01	lbfgs
1	3	0.593301818	0.604664433	tanh	0.1	(2,)	0.001	lbfgs
1	4	0.593281768	0.604676167	tanh	0.1	(2,)	0.1	lbfgs
1	5	0.593281274	0.605039433	logistic	0.01	(2,)	0.01	lbfgs
1	6	0.59303834	0.605053696	logistic	0.01	(2,)	0.1	lbfgs
1	7	0.593007103	0.605412671	logistic	1.00E-05	(2,)	0.01	lbfgs
1	8	0.592890816	0.605319711	logistic	0.001	(2,)	0.1	lbfgs
1	9	0.592734906	0.605364771	tanh	1.00E-05	(2,)	0.1	lbfgs
1	10	0.592667488	0.605400078	tanh	0.0001	(2,)	0.1	lbfgs
2	1	0.58908515	0.599173606	logistic	1.00E-05	(2,)	0.001	lbfgs
2	2	0.588914767	0.599235334	tanh	1.00E-05	(2,)	0.01	lbfgs
2	3	0.588890983	0.59922873	tanh	0.001	(2,)	0.1	lbfgs
2	4	0.588355615	0.598930748	logistic	1.00E-05	(2,)	0.1	lbfgs
2	5	0.5881249	0.598810318	logistic	1.00E-05	(2,)	0.01	lbfgs
2	6	0.588122543	0.598896281	logistic	0.001	(2,)	0.01	lbfgs
2	7	0.588010809	0.598769971	logistic	0.001	(2,)	0.001	lbfgs
2	8	0.587951581	0.598800996	logistic	0.0001	(2,)	0.001	lbfgs
2	9	0.587946902	0.598971598	logistic	0.0001	(2,)	0.1	lbfgs
2	10	0.587838655	0.598805757	logistic	0.001	(2,)	0.1	lbfgs
3	1	0.591649157	0.601265344	logistic	1.00E-05	(2,)	0.01	lbfgs
3	2	0.591540179	0.606091557	logistic	0.001	(3,)	0.1	lbfgs
3	3	0.591538304	0.601528831	tanh	1.00E-05	(2,)	0.001	lbfgs
3	4	0.591465217	0.601223053	logistic	0.0001	(2,)	0.1	lbfgs
3	5	0.591405312	0.601230329	tanh	0.001	(2,)	0.001	lbfgs
3	6	0.591399387	0.601143429	logistic	0.001	(2,)	0.1	lbfgs

3	7	0.591347082	0.601217297	logistic	0.0001	(2,)	0.001	lbfgs
3	8	0.591341633	0.601126851	logistic	0.001	(2,)	0.001	lbfgs
3	9	0.591317352	0.601203535	logistic	1.00E-05	(2,)	0.1	lbfgs
3	10	0.591313697	0.605725764	tanh	0.001	(3,)	0.1	lbfgs
4	1	0.583905121	0.597120345	logistic	0.001	(3,)	0.001	lbfgs
4	2	0.582009151	0.597570737	tanh	0.01	(3,)	0.001	lbfgs
4	3	0.581996175	0.593885881	logistic	0.0001	(2,)	0.01	lbfgs
4	4	0.581820397	0.593897002	tanh	0.001	(2,)	0.001	lbfgs
4	5	0.581766082	0.593907486	logistic	0.0001	(2,)	0.1	lbfgs
4	6	0.581642362	0.593912615	logistic	1.00E-05	(2,)	0.001	lbfgs
4	7	0.581607776	0.594294767	logistic	1.00E-05	(2,)	0.1	lbfgs
4	8	0.581539611	0.593858119	logistic	0.001	(2,)	0.1	lbfgs
4	9	0.581505448	0.598127591	logistic	0.001	(3,)	0.1	lbfgs
4	10	0.581458361	0.593844988	logistic	0.001	(2,)	0.001	lbfgs

Table 3 shows the top 5 inner-loop mean R^2 score out of all different hyperparameter combinations used when tuning a neural network for each outer-loop fold. To keep the report brief, the rest of the results are available in data/report_analysis/nn_i.csv, where i refers to an outer-loop fold number.

Neural Network Fold 0: The final hyperparameters chosen were solver=lbfgs, activation=tanh, alpha=1e-05, learning_rate_init=0.01, and hidden_layer_sizes=(3,); the mean validation score of these chosen values was 0.5861069818899927 and mean training score was 0.599733948. Total computation time necessary to run the inner-loop experiment was 707.391942977905 seconds.

Neural Network Fold 1: The final hyperparameters chosen were solver=lbfgs, activation=tanh, alpha=0.1, learning_rate_init=0.01, and hidden_layer_sizes=(2,); the mean validation score of these chosen values was 0.5933676006111376 and mean training score was 0.604681557. Total computation time necessary to run the inner-loop experiment was 758.017703294754 seconds.

Neural Network Fold 2: The final hyperparameters chosen were solver=lbfgs, activation=logistic, alpha=1e-05, learning_rate_init=0.001, and hidden_layer_sizes=(2,); the mean validation score of these chosen values was 0.5890851497297958 and mean training score was 0.599173606. Total computation time necessary to run the inner-loop experiment was 749.2678713798525 seconds.

Results continue on next page.

Neural Network Fold 3: The final hyperparameters chosen were solver=lbfgs, activation=logistic, alpha=1e-05, learning_rate_init=0.01, and hidden_layer_sizes=(2,); the mean validation score of these chosen values was 0.5916491565501066 and mean training score was 0.601265344. Total computation time necessary to run the inner-loop experiment was 789.2151975631715 seconds.

Neural Network Fold 4: The final hyperparameters chosen were solver=lbfgs, activation=logistic, alpha=0.001, learning_rate_init=0.001, and hidden_layer_sizes=(3,); the mean validation score of these chosen values was 0.5839051214742994 and mean training score was 0.597120345. Total computation time necessary to run the inner-loop experiment was 743.2922644615172 seconds.

IV. Comparing Algorithm Performance

Experiment: After hyperparameters are tuned, the unseen testing set in each fold of the 5-fold cross-validation (outer-loop) will be used to score the trained model. The R^2 score 95% confidence interval of all 5 outer-loop folds will be used to compare algorithm performance.

Table 4: 5-Fold Cross-Validation Algorithm Performance

Algorithm	Testing Mean	Testing Standard Deviation	95% Confidence Interval Lower Bound	95% Confidence Interval Upper Bound
Kernel Ridge Regression	0.5717053349404521	0.019005418824219064	0.5550463508351834	0.5883643190457207
k-Neighbors Regression	0.5361691967929969	0.02521017421292875	0.5140715047932263	0.5582668887927675
Neural Network	0.5892111127462716	0.017519463074066377	0.573854626284011	0.6045675992085322

Based on the results, Neural Networks had the best performance on the dataset. Neural Networks had the highest testing mean and the smallest testing standard deviation out of all of the algorithms; the 95% confidence interval for Neural Networks is the only one to reach a score of 60% or higher. The 95% confidence interval lower bound for Neural Networks is higher than the 95% confidence interval upper bound for k-Neighbors Regression and closer to the 95% confidence interval upper bound for Kernel Ridge Regression than it is to the 95% confidence interval lower bound for Kernel Ridge Regression.

V. Conclusion

k-Neighbors Regression had a shorter computation time relative to the other algorithms; Neural Networks had a longer computation time relative to the other algorithms. Kernel Ridge Regression and k-Neighbors Regression was tested on three hyperparameters and Neural Networks were tested on five hyperparameters. Based on computation time, number of hyperparameters, and performance, Kernel Ridge Regression would be better than the other two algorithms to use in practice on real world data; Kernel Ridge Regression provides a faster computation time with relatively decent performance and less hyperparameters necessary to tune.

VI. Acknowledgments

Asuncion, A. & Newman, D.J. (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Wes McKinney (2010). Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference (pp. 51 - 56).