Auto Machine Learning for Predicting Wine Quality.

Melissa Butler and Emma Franz
Department of Mathematics and Statistics
University of Wyoming, Laramie, WY

COSC 5010 - Practical Applications of Machine Learning - Spring 2023
University of Wyoming
May 04, 2023

Introduction

The *Franz Butler Vineyard*™ would like to predict how much they can charge for a bottle of wine from their vineyard. The local market has three main price points: Box, Good, and Fancy. A dataset of quantitative descriptions of 1599 red wines was obtained from `https://archive-beta.ics.uci.edu/dataset/186/wine+quality`.

- Six machine learning classifers were compared over a 6-category split vs. a 3-category split.
- The hyperparameters and classifiers were then optimized using Bayesian Optimization.
- Finally a random search was used to optimize a pipeline including classifiers, hyperparameters, data scaling, and feature selection.

Data

The input values are 11 physiochemical properties with numeric ratings.

	mean	std	min	max
fixed acidity	8.32	1.74	4.600	15.900
volatile acidity	0.53	0.18	0.120	1.580
citric acid	0.27	0.19	0.000	1.000
residual sugar	2.54	1.41	0.900	15.500
chlorides	0.09	0.05	0.012	0.611
free sulfur dioxide	15.87	10.46	1.000	72.000
total sulfur dioxide	46.47	32.90	6.000	289.000
density	1.00	0.00	0.990	1.004
pH	3.31	0.15	2.740	4.010
sulphates	0.66	0.17	0.330	2.000
alcohol	10.42	1.07	8.400	14.900

Figure: Features

Data

The ouput to be predicited is a quality rating, from sensory data ranging from 3-8. We also explored a 3 subcategory grouping, to reflect market princing:

$$\text{Box} = \{3, 4\}, \quad \text{Good} = \{5, 6\}, \quad \text{Fancy} = \{7, 8\}$$

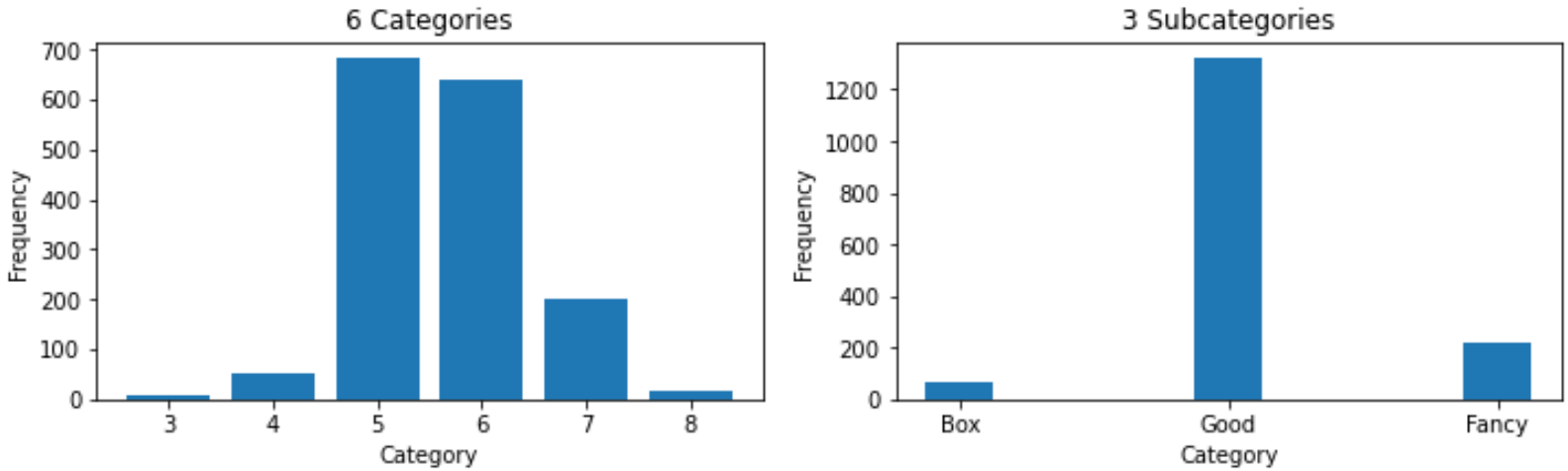

Figure: Category Frequency

Classifiers with Default Values

- Six estimators were tested using *scikit-learn*: a Decision Tree classifier, a Random Forest classifier, an AdaBoost classifier, a K-nearest neighbors classifier, a Support Vector classifier, and a Support Vector Regression.
- The cost function we seek to maximize is accuracy. For the Support Vector Regression, we define accuracy to be the proportion of predicted values that, once rounded to the nearest integer, match the test values.
- A k-fold Cross Validation with k $= 10$ was used to ensures that each data point is used in both testing and training and prevent overfitting to a single test-train split.
- The data was normalized by feature using the L2 norm.

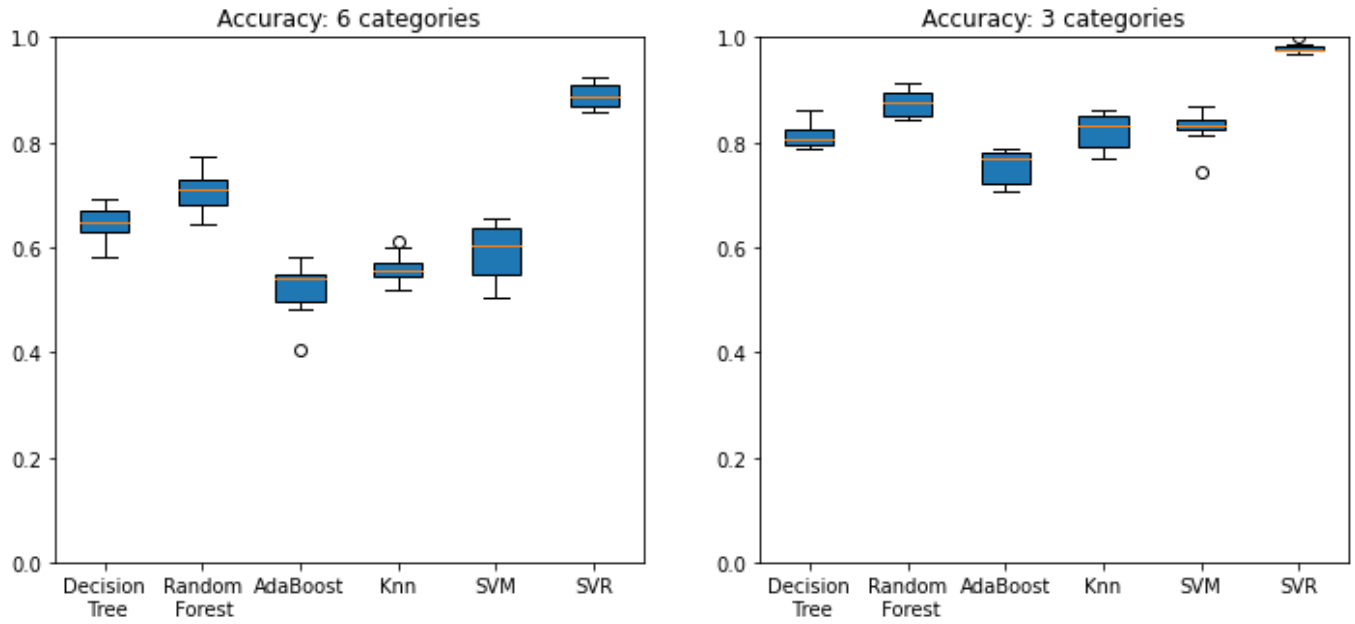

Figure: Estimator Optimization (Default Hyperparameter Values)

- Three Categories: All estimators perform well and we could stop here.
- Six Categories:
  - Best Performing Estimator: Support Vector Regression $\sim 88.9\%$ accuracy.
  - Best Performing Classifier: Random Forest $\sim 69.8\%$

Hyperparameter Optimization using BayesSearchCV()

- The BayesSearchCV() function from *scikit-optimize* was used to search a space of classifiers and their corresponding hyperparameter ranges.
- We used a 3 by 3 nested resampling to first determine a best hyperparameter setting for each estimator, then generate an unbiased performance estimate of this hyperparameter setting.
- The accuracy scoring and data normalization are as before.
- For each estimator, the number of iterations for the Bayes search was 50.

Hyperparameter ranges for BayesSearchCV()

Classifier	Hyperparameter	Default	Range
Decision Tree	max features	None	log or sqrt(# of features)
	min samples/leaf	1	$2 \leq i \leq 10$
	min samples/split	2	$2 \leq i \leq 10$
Random Forest	n estimators	100	100, 500
	max features	None	log or sqrt(# of features)
	min samples/leaf	1	$2 \leq i \leq 10$
	min samples/split	2	$2 \leq i \leq 10$
KNN	n neighbors	5	$1 \leq i \leq 10$
	weights	uniform	uniform, distance
	algoritm	auto	auto, ball tree, kd tree, brute
Ada Boost	n estimators	50	$1 \leq i \leq 1000$
	learning rate	1	$0.01 \leq x \leq 10.0$
SVC, SVR	kernel	rbf	linear, poly, rbf, sigmoid
	C	1	$0.01 \leq x \leq 1000.0$
	gamma	scale	$0.01 \leq x \leq 1000.0$
	degree	3	$2 \leq i \leq 7$

Hyperparameter Optimization Results

FOLD	DT	RF	Ada	KNN	SVM	SVR
1	0.553	0.675	0.567	0.612	0.585	0.887
2	0.523	0.672	0.567	0.612	0.612	0.895
3	0.505	0.664	0.523	0.638	0.597	0.865

Figure: Testing Accuracy

FOLD	DT	RF	Ada	KNN	SVM	SVR
1	0.579	0.656	0.567	0.612	0.600	0.886
2	0.586	0.651	0.565	0.627	0.600	0.881
3	0.585	0.667	0.581	0.605	0.614	0.899

Figure: Training Accuracy

	DT	RF	Ada	KNN	SVM	SVR
Default HP	0.625	0.698	0.524	0.568	0.58	0.889

Figure: Default Hyperparameters

Complete Pipeline

Adding additional components to the pipeline while still guaranteeing unbiased testing required a more streamlined pipeline. For this we used Pipeline from *scikit-learn* and Pipelinehelper from `https://github.com/bmurauer/pipelinehelper`.

- Data scaling and feature selection were added
- Hyperparameter ranges were converted to discrete values
- RandomSearchCV() from *scikit-learn* was used

Data Scaling

Three data scaling options were included in the search space, to assist the distance based classifier KNN.

- Min-Max : all feature values are positive with varying ranges
- Absolute-Max : all feature values are positive with varying ranges
- Robust : Account for any outliers
  - Quartile ranges: $(25, 75), (15, 75), (25, 85)$

Values are not assumed to be normally distributed, so a standard scalar was not used. Other classifiers are decision based and do not benefit from scaling.

Feature Selection

To improve computation time and reduce cost of testing future wine, feature selection is implemented.

- Variance Threshold - correlation between features
  - $0.1, 0.5, 0.9$ - current literature seems to use a guess and check method
  - none - include all features
- Univariate Selection - most influential features
  - ANOVA F-Test Classification - target is categorical and features are continous
  - Select K Best: $3, 9, 12$ - the vineyard's current lab has these three pricing options

New Hyperparameter Values

Discrete hyperparameter values were chosen from previous ranges, as well as the default values.

Classifier	Hyperparameter	Default	Range
Random Forest	n estimators	100	100, 500
	max features	None	1, sqrt(# of features)
K-Nearest Neighbors	n neighbors	5	1,2,3,4,5,6,7,8,9,10
	weights	uniform	uniform, distance
Ada Boost Classifier	n estimators	50	10, 50, 100
	learning rate	1	0.01, 1, 5, 10
Support Vector Classifier	kernel	rbf	rbf, linear
	C	1	0.1, 1, 10, 100, 1000
Support Vector Regression	kernel	rbf	rbf, linear
	C	1	0.1, 1, 10, 100, 1000

Results: cv=3, 50 iterations

```
Fitting 3 folds for each of 50 candidates, totalling 150 fits
RandomizedSearchCV took 36.65 seconds for 50 candidates.
Model with rank: 1
Mean validation score: 0.885 (std: 0.018)
Parameters: {'scaler__selected_model': ('robust', {'quantile_range': (25.0, 75.0)}),
             'features__selected_model': ('kbest', {'k': 9}),
             'classifier__selected_model': ('svr', {'C': 0.1, 'kernel': 'rbf'})}

Model with rank: 2
Mean validation score: 0.882 (std: 0.019)
Parameters: {'scaler__selected_model': ('robust', {'quantile_range': (25.0, 85.0)}),
             'features__selected_model': ('kbest', {'k': 9}),
             'classifier__selected_model': ('svr', {'C': 0.1, 'kernel': 'rbf'})}

Model with rank: 3
Mean validation score: 0.876 (std: 0.014)
Parameters: {'scaler__selected_model': ('max', {'copy': True}),
             'features__selected_model': ('kbest', {'k': 9}),
             'classifier__selected_model': ('svr', {'C': 0.1, 'kernel': 'linear'})}
```

Conclusion

- A Support Vector Regression model is the most promising model in all cases.
- We could possibly increase accuracy by:
  - Increasing ranges on Hyperparameters (robust scaling and SVR C-value)
  - Employing Bayesian Optimization
- Invest in a fancy bottle and high end marketing campaign to sell any wine as Fancy.

Resources

https://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html#
sphx-glr-auto-examples-model-selection-plot-randomized-search-py
https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35
https://www.kaggle.com/code/tanmayunhale/feature-selection-variance-threshold
https://towardsdatascience.com/mistakes-in-applying-univariate-feature-selection-methods-34c43ce8b93d
https://github.com/bmurauer/pipelinehelper https://scikit-learn.org/stable/index.html

https://github.com/Emmafranz/PAML_1.git

THANK YOU