MACHINE LEARNING : WHITE QUALITY WINE

AUTHORS: AHMED MOHAMED – MOHAMED BHAILAT

CITY UNIVERSITY LONDON

1. MOTIVATION AND DESCRIPTION

Motivation:

. There has been a lot of challenges in the wine industry to differentiate and improve the quality of the wine. Usually, wine is being tested in a physicochemical lab to try to characterize the chemical variables of the content to improve the quality of wine. In a contribution to that using K-Nearest Neighbor and Random Forest algorithms, our models will attempt to predict the wine quality based on physicochemical properties using white wine UCI dataset. Such an attempt will help wine producers to improve winemaking in the future.

. Our results will be compared with previous intuitive approach by (Atasoy & Er , 2016) [6].

Data Description:

.Dataset: White wine dataset from UCI, it has 11 numerical predictors (physicochemical properties), 1 numerical label (quality class on a scale 3-9, where 3 poor and 9 excellent) and 4899 entities.

.There is a class imbalance as seen in Figure 3, which shows the distribution of wine quality, class 3 and 9 have few data points which will not be sufficient in building a model. Also, Imbalance can affect the model by achieving high accuracy by favoring the majority class. In response to this, the data were resampled assuming that class 6 is our average wine and anything below 6 is poor and anything above 6 is good in order to have enough data, which is shown in Figure 4.

.From the correlation heat map, as shown in Figure 2, it could be deduced that there is some correlation between physicochemical variables and the quality.

. The statistics summary table (see Figure 1) shows the calculated mean and standard deviation for each attribute in the three classes, it shows as well some interesting relationship such as good wine has more alcohol and low chlorides.

	Maximum			Minimum			Mean			Std		
Feature	Bad	Medium	Good	Bad	Medium	Good	Bad	Medium	Good	Bad	Medium	Good
Fixed acidity	11.80	14.20	9.20	4.20	3.80	3.90	6.96	6.84	6.73	0.88	0.84	0.77
Volatile acidity	1.10	0.97	0.76	0.10	0.08	0.08	0.31	0.26	0.27	0.11	0.09	0.09
Citric acid	1	1.66	0.74	0.00	0.00	0.01	0.33	0.34	0.33	0.14	0.12	0.08
Residual sugar	23.50	65.80	19.25	0.60	0.70	0.80	7.05	6.44	5.26	5.28	5.16	4.29
Chlorides	0.35	0.26	0.14	0.01	0.02	0.01	0.05	0.05	0.04	0.03	0.02	0.01
Free sulfur dioxide	289	112	108	2	3	5	35.34	35.65	34.55	20.22	15.74	13.80
Total sulfur	440	294	229	9	18	34	148.60	137.03	125.25	46.91	41.29	32.72
Density	1.00	1.04	1.00	0.99	0.99	0.99	1.00	0.99	0.99	0.00	0.00	0.00
pH	3.79	3.81	3.82	2.79	2.72	2.84	3.17	3.19	3.22	0.14	0.15	0.16
Sulphate	0.88	1.06	1.08	0.25	0.23	0.22	0.48	0.49	0.50	0.10	0.11	0.13
Alcohol	13.60	14.00	14.20	8.00	8.50	8.50	9.85	10.58	11.42	0.88	1.15	1.26

Figure 1: Statistics summary table of the dataset

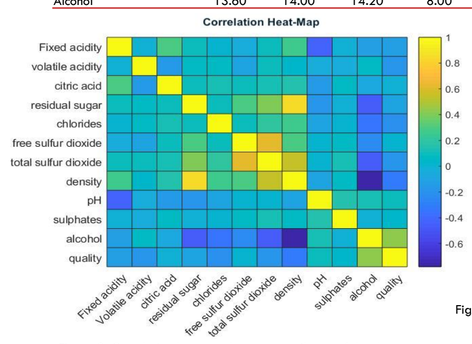

Figure 2: Correlation heatmap between attributes of the dataset

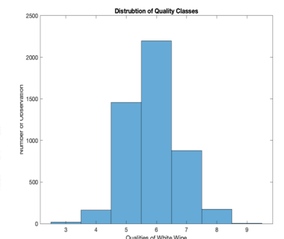

Figure 3: Histogram showing the imbalance in the label

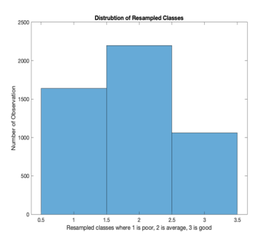

Figure 4: Histogram showing the resampled label

2. RANDOM FOREST OVERVIEW

Algorithm Description:

. Random Forest is a classifier that's composed of a collection of constructed classifiers trees on bootstrap of data samples[1]; which could be expressed as follows:

$$\{h(\mathbf{x},\Theta_k^{'}), k=1, ...\}$$

. Where k is the number of trees which has randomly distributed vectors represented in O_k and the trees are being developed using O_k and a bagged subset of the training set. Therefore, resulting in $h(x,O_k)$ classifier, where each individual tree put a vote for the most favored class at input x.

Algorithm Pros:

1. Random Forest is insensitive to overtraining and noise due to the resampling process.
2. Minimizing variance between classifiers by looking only at random variables subsets in the nodes without increasing bias[2].
3. RF could model classification and regression problems and it works robustly without preprocessing as it handles outliers, missing values[2].
4. It has low chance of overfitting data due to the distinct nature of O_k [3].
5. It works perfectly with small and large datasets, it could also be used for detecting variable interaction.

Algorithm Cons:

1. The computational time could significantly slow if the number of trees to be grown required is a lot.
2. It is prone when dealing with categorical data level as it tends to favor the high–level attributes[3].
3. Minimal control of the model, and could be time-consuming in optimizing hyperparameter.

3. K-NEAREST NEIGHBOR OVERVIEW

Algorithm Description:

. K-Nearest Neighbor algorithm is non-parametric method that computes the distance between the point to be predicted and number of nearest training examples[4]. It could be expressed as follows:

$$x^i = (x^i_1, x^i_2, \ldots, x^i_p) \qquad d_E(x^i, x^j) = \left(\sum_{k=1}^{p} (x^i_k - x^j_k)^2\right)^{\frac{1}{2}}$$

. Where x^i here represent an attribute with p measurements and x^j is the predictor, where the distance could be measured using different functions such as Euclidean distance (d_E) like shown above.

Algorithm Pros:

1. No assumption about the data, the only assumption is proximity, due to the fact that it is a non-parametric method so no distribution fitting is required[5].
2. The building model could be easily applied.
3. It works robustly with overlapping data as it stores all the training data.

Algorithm Cons:

1. It could crash in high dimensional space as the neighborhood could become significantly high.
2. Increasing the K will minimize the variance but will increase bias[5].
3. Sensitive to outliers, missing values and irrelevant attributes, so it require more pre-processing than Random Forests[4].
4. The algorithm is not actually learning, it is storing all the training instances and then doing comparison, which makes it computationally expensive, as time would be expressed as (n*D) where n is training instances number and D is computing distance[4].

4. HYPOTHESIS STATEMENT

.It's expected to our prior knowledge that Random Forest will perform more robustly than KNN.

.As a previous empirical study reported that Random Forest and KNN have validation error of 0.2963 and 0.35 respectively[6].

.Also, after considering the pros and cons of each model in the above section, it's expected that Random Forest would perform better.

5. TRAINING AND EVALUATION DESCRIPTION

. The dataset was split after being resampled in the initial investigation (see Figure 4) into an 80% training data and 20% test data.

. Two optimization methods were performed upon both models to choose the parameters that result in the least error.

. The dataset was normalised then validated with 10-fold cross-validation to avoid overfitting.

. DFO algorithm (discussed in extra work) was performed in an attempt to find the best features that optimize results, but it was discarded.

.The accuracy error was used a main evaluation model metric in comparing models then AUC-ROC curve was also used in analysis part.

6. RANDOM FOREST OVERVIEW

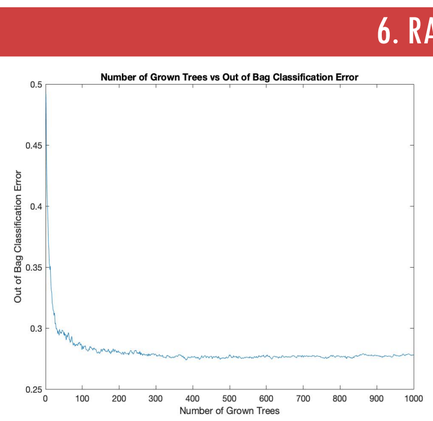

Figure 5: A graph showing the OOB error against the number of trees

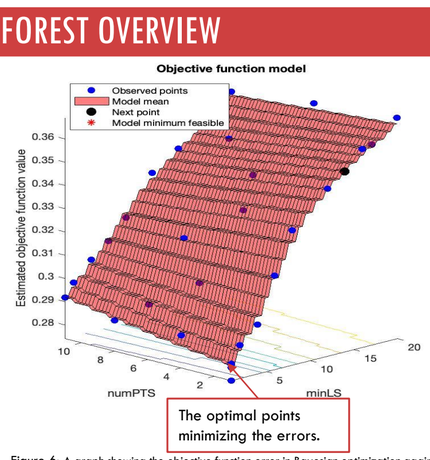

Figure 6: A graph showing the objective function error in Bayesian optimization against the number of predictors sampled at each node(numPTS) and minmum number of observation perf leaf (minLS)

Parameters Tuning :

. Bayesian optimization and Grid search were run on Random Forest to find the most optimal:

- Minimum number of observation per leaf.
- Number of predictors to sample at each node.

Results:

. The most optimal parameters using Bayesian optimization and 1000 trees, shown in Figure 6, were 1 feature sampled at each node and a minimum leaf size of 1, resulting in 27.93% error.

. By using Grid search and 1000 trees, the optimal parameters were 1 feature sampled at each node and a minimum leaf size of 2, resulting in 27.35% error.

. The accuracy decreased as the number of features sampled and minimum leaf size increased.

. On the other hand, the out of bag error decreased (OOB error) with a high number of trees grown, as shown in Figure 5, but it's a trade-off as computational time increases.

. The final model was 72.32% accurate and took 40.73s to build.

7. K-NEAREST NEIGHBOR OVERVIEW

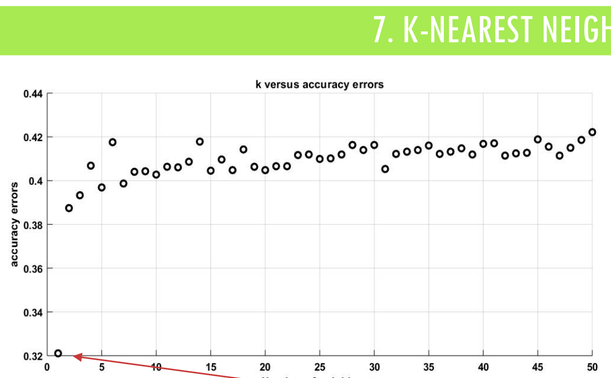

Figure 7: A graph showing the objective function error against number of neighbors using grid search

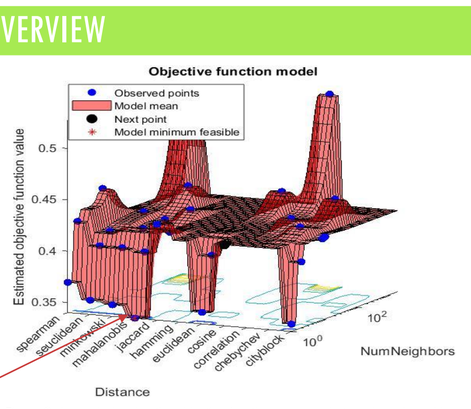

Figure 8: A graph showing the objective function error against number of neighbors using and distance function using Bayesian optimization

KNN having the least accuracy error with K=1 and Mahalanobis.

KNN		Random Forest
32.11%	Cross Val Training	27.35%
32.99%	Test	27.68%
5.02s	Time	40.73s

Parameters Tuning :

. Bayesian optimizations and Grid search were run to try to find the most optimal:

- K: the number of neighbors in the object space.
- The distance function.

Results:

. The most optimal parameters when using Bayesian optimization and Grid search was k=1 and Mahalanobis as a distance function which results in 32.11% error, which is shown in Figure 8.

. The error increased relatively as the number of neighbors got higher than k=1 which is shown in Figure 7.

. The final model was 67.01% accurate which is less than Random Forest but was computationally lighter and took 5.20s to build.

8. TRAINING AND EVALUATION DESCRIPTION

1. KNN optimal number of neighbors is 1, which results in significantly low bias and high variance. Also, it indicates that we are fitting near training dataset and we don't have a desirable smooth function as there will significant noise and distortion.
2. The optimal KNN model was k=1 as it could indicate that our 3 classes might be linearly separable in the space. However, such a model would not be ideal to generalize for new data. In another approach, the parameters grid search was limited from 3 to 50 and shows that k=9 is the optimal number of neighbors if a trade-off to be considered between accuracy and overfitting.
3. The Grid search outperformed the Bayesian optimization, as our objective function was not computationally expensive. However, if it is expensive Bayesian is better as it is an approximation based on a surrogate probability model of objective function.
4. Random Forest performed better in validation and test accuracy than KNN, it could also be deduced also that Random Forest less prone to overfitting the data due to the bagging and boosting which results in its discrete nature of minimizing the variance without being bias.
5. The Random Forest model took 8 times more than KNN as there 1000 trees grown due to the fact it's more accurate as the number of trees increase.
6. The optimal number of predictors sampled at each node is 1 but if increased ,our RF model becomes less accurate as features redundancy on each node would result in identical correlated trees. So this added correlation will surpass the gain in strength of trees maximizing the following function $\frac{C(correlation)}{S^2(Tree\ Strength)}$ which will increase the error according to Breiman[1].
7. KNN would be performing better if we had bigger training examples and that is one of the reasons that we resampled our data in the initial investigation.

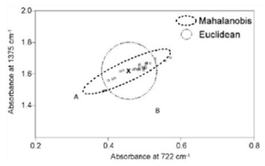

Figure 9: Conversion from Euclidean function to Mahalanobis[9]

8. The optimal distance function in KNN is Mahalanobis, that was expected due to a prior knowledge of some correlation between our variables such as alcohol and quality as Mahalanobis takes the variables correlation into account. Nevertheless, Mahalanobis requires inversion of covariance matrix (see Figure 9) which could be computationally expensive but it was not an obstacle in our case[9].

Figure 10: AUC-ROC curve of three classes against each other under the performance of KNN and RF.

10. When comparing KNN and Random Forest using AUC-ROC curve, to show which model classified the 3 classes, it shows that RF classified bad and medium quality wine better than KNN. On the other, KNN outperformed RF in classifying the good wine.
11. It's an interesting that KNN classifies the good wine (class) better as it actually indicates that Random Forest suffers from imbalanced classes and favor the one with high observation[11], which is also stated in (Li et al., 2011) as the good class has less observation than other classes.
12. The average AUC score of KNN and Random Forest are 76.05% and 76.38% respectively, which is a good conclusion that accuracy measure might still be affected by the class imbalance in our dataset. As accuracy could misleading in model evaluation when classes are imbalanced[10].
13. In conclusion, referring back to the hypothesis, RF performed better like our expectation but its performance was surprisingly bad in the imbalanced class. However, our result using KNN was higher than the previous study done as they used the second-best number of neighbors and avoided k=1.

9. EXTRA AND FUTURE WORK

Extra Work Done:

.In another approach, DFO, an algorithm that optimizes the problem by searching broad space of candidate solutions[7]. It was used to search for the most optimal features. Its an ideal solution as KNN could crash in high dimensions, however due to the size of our dataset it did not boost the performance of our models.

Future Work:

. It would be interesting to find methods that could optimize the k but considers as well balancing the variance and bias.

. Removing outliers as well would a be a good approach to boost the KNN performance.

. Finally, balancing the data using Synthetic Minority Over-Sampling Technique should be considered[8].

10. REFERENCES

1. Breiman, L., 2001. Random forests. *Machine learning*, *45*(1), pp.5-32.
2. Ravanshad, A. (2015). Gradient Boosting vs Random Forest. [online] Medium.com. Available at: https://medium.com/@aravanshad/gradient-boosting-versus-random-forest-cfa3fa8f0d80 [Accessed 11 Nov. 2018].
3. Gislason, P.O., Benediktsson, J.A. and Sveinsson, J.R., 2006. Random forests for land cover classification. *Pattern Recognition Letters*, *27*(4), pp.294-300.
4. Song, Y., Huang, J., Zhou, D., Zha, H. and Giles, C.L., 2007, September. Iknn: Informative k-nearest neighbor pattern classification. In European Conference on Principles of Data Mining and Knowledge Discovery (pp. 248-264). Springer, Berlin, Heidelberg.
5. Goddard, N. (2016). Pros and cons of nearest-neighbor methods. [online] Media Hopper Create. Available at: https://media.ed.ac.uk/media/Pros+and+cons+of+nearest-neighbor+methods/1_ahjryv3t [Accessed 11 Nov. 2018].
6. Er, Y. and Atasoy, A., 2016. The classification of white wine and red wine according to their physicochemical qualities. International Journal of Intelligent Systems and Applications Engineering, 4, pp.23-26.
7. Al-Rifaie, M.M., 2014, September. Dispersive flies optimisation. In Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on (pp. 529-538). IEEE.
8. Hu, G., Xi, T., Mohammed, F., & Miao, H. (2016). Classification of wine quality with imbalanced data. 2016 IEEE International Conference on Industrial Technology (ICIT). doi:10.1109/icit.2016.7475021
9. Walters-Williams, J. and Li, Y., 2010. Comparative study of distance functions for nearest neighbors. In Advanced Techniques in Computing Sciences and Software Engineering(pp. 79-84). Springer, Dordrecht.
10. Akosa, J., 2017. Predictive Accuracy: A Misleading Performance Measure for Highly Imbalanced Data. In Proceedings of the SAS Global Forum.
11. Li, S., Harner, E.J. and Adjeroh, D.A., 2011. Random KNN feature selection-a fast and stable alternative to Random Forests. *BMC bioinformatics*, *12*(1), p.450.