Date: 19.04.2023

Author: Mughdat Hajizada

Model Selection

Research questions: *"How accurately can we predict the quality of wine based on its properties using various regression models? Which model is performing better?"*

Contents:

1. Data Description

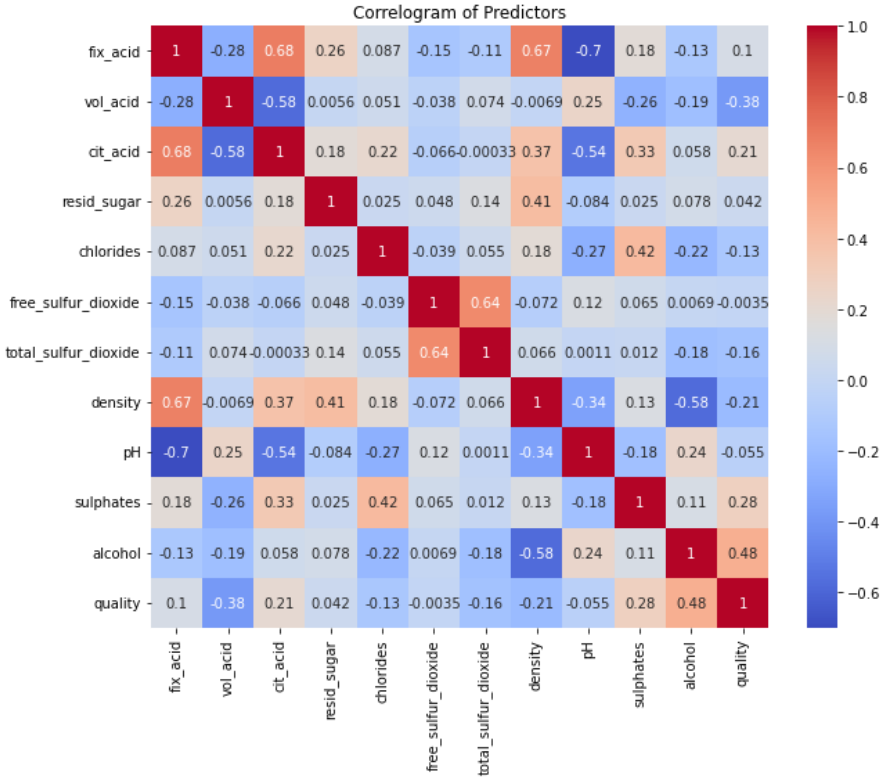
The primary objective of this paper is to identify the most effective regression models applicable to the "Red Wine Quality" dataset (Cortez et al., 2009). The shape of the data is (1599, 12) which means there are 1599 observations and 12 attributes. Dataset is clean – there are no NA values. All columns converted into a float.

Here are basic descriptive statistics of the whole dataset:

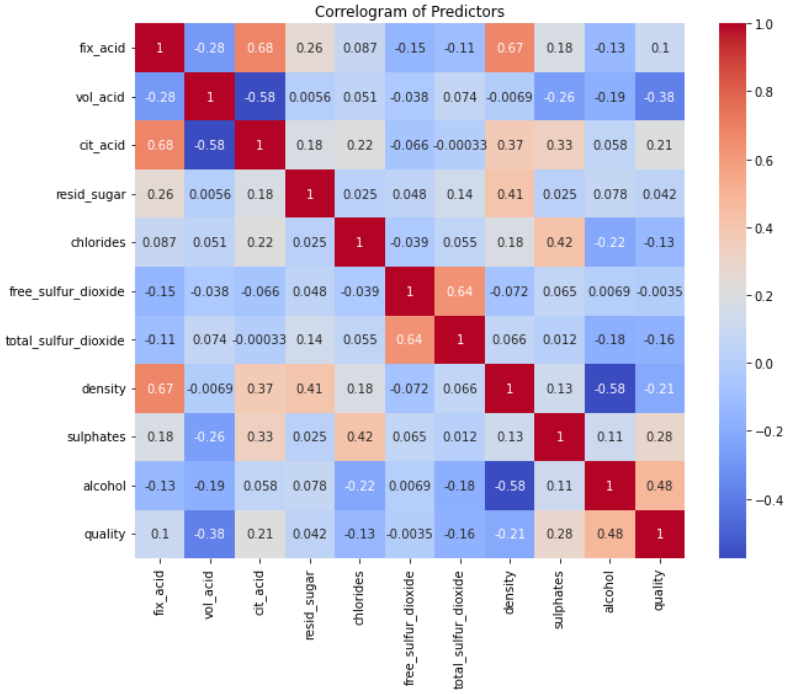
Statistics	fix_acid	vol_acid	cit_acid	resid_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alcohol	quality
count	1599	1599	1599	1599	1599	1599	1599	1599	1599	1599	1599	1599
mean	8,320	0,528	0,271	2,539	0,087	15,875	46,468	0,997	3,311	0,658	10,423	5,636
std	1,741	0,179	0,195	1,410	0,047	10,460	32,895	0,002	0,154	0,170	1,066	0,808
min	4,6	0,12	0	0,9	0,012	1	6	0,99007	2,74	0,33	8,4	3
25%	7,1	0,39	0,09	1,9	0,07	7	22	0,9956	3,21	0,55	9,5	5
50%	7,9	0,52	0,26	2,2	0,079	14	38	0,99675	3,31	0,62	10,2	6
75%	9,2	0,64	0,42	2,6	0,09	21	62	0,997835	3,4	0,73	11,1	6
max	15,9	1,58	1	15,5	0,611	72	289	1,00369	4,01	2	14,9	8

It can be observed from the above table that attributes named *resid_sugar, total_sulfur_dioxide, free_sulfur_dioxide* may contain outliers. As outliers can have disproportionate impact on model performance, it is recommended to get rid of them by creating upper boundary for outliers using formula:

$$(Q3-Q1)*1.5+Q3$$



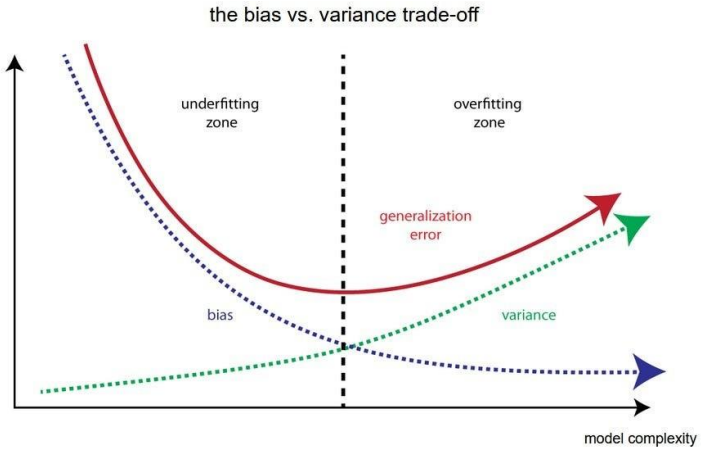
Looking at the above correlogram of the dataset, it can be observed that attribute *pH* has -0.7 degree of correlation with *fix_acid*. To avoid multicollinearity problem, attribute pH will be dropped from the dataset. After dropping pH, no other predictors have a strong relationship with other predictors (see below graph).



After dropping outliers and pH variable, the shape of the dataset is (r-1390, c-11).

2. Variable Selection

This is an important part of every model selection. It is crucial to define which predictors contribute more to the performance of the models. Adding more predictors into the model can cover the huge part of variability of the data but it may lead to overfitting. In other words, model will perform well in training model, but the performance will be worse for the live data. On the contrary, making models less complex will cause extra bias. This is called bias-variance tradeoff and the goal is to optimize the model considering both bias and variance.



There are different ways to define optimal predictors. In this example, the Lasso CV method will be used. Lasso is a method of regularization that involves adding a penalty term to the cost function, leading the coefficients towards zero, thereby reducing overfitting. One of the benefits of using Lasso is that it selects variables automatically by assigning some of the coefficients to zero, which is helpful when working with high-dimensional data where not all the predictors are relevant. Cross-validation will be used to find the optimal regularization parameter and predictors.

Output of the Lasso CV is the following:

Optimal number of predictors: 7

Optimal alpha value: 0.0034355

Selected features: *fix_acid, vol_acid, cit_acid, free_sulfur_dioxide, total_sulfur_dioxide, sulphates, alcohol*

After selecting features, the dataset will be split into training, test, and holdout sets. Then, the models will be trained, and their performance will be evaluated using cross-validation on the training set. The best-performing models will be selected by tuning their parameters to optimize their performance. Finally, the selected models will be evaluated on the test and then on holdout sets to estimate their true performance on unseen data.

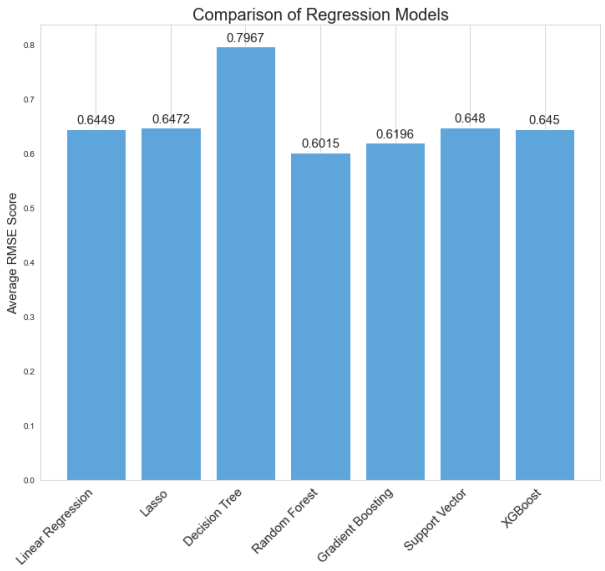
3. Model Selection

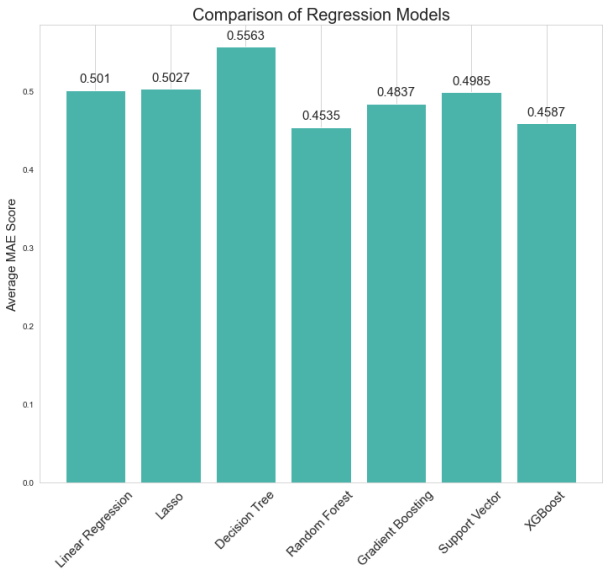
Model selection is crucial in data analysis, as it involves choosing the best model out of several alternatives that fit the data. It should be able to generalize well and avoid overfitting. Proper evaluation and comparison of different models using appropriate metrics is necessary before making the final decision. To get more accurate metrics, cross validation will be used to evaluate the performance of different models on multiple subsets of the data.[1]

Models: *Linear Regression, Lasso, Decision Tree Regressor, Gradient Boosting Regressor, Random Forest Regressor, Support Vector Regression, XGB Regressor*

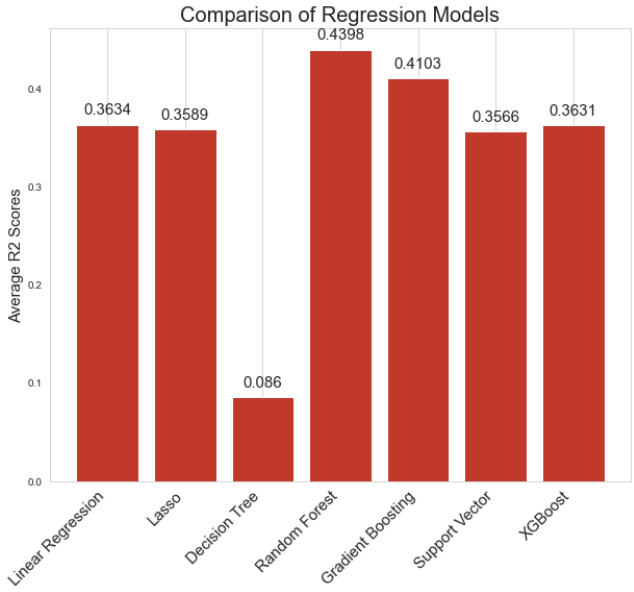
Metrics: *RMSE, MAE, R-squared*

Model performances can be seen from the below graphs:





[1] There are few changes for default parameters of several models (DTR, SVR, XGB) to improve their performance. Others have default settings.



Upon examining the graphs presented, it is evident that the Random Forest model outperforms the other models. This can be attributed to its significantly lower values for both RMSE and MAE, as well as its higher R-squared score. The Gradient Boosting model can be considered the second best-performing model, as it demonstrates superior performance when compared to all other models with respect to lower RMSE and higher R-squared scores. The only model that surpasses the Gradient Boosting model in terms of MAE is XGBoost, while the Gradient Boosting model outperforms the remaining models.

4. Optimizing Parameters

The current approach involves identifying the two best models using their default settings, and subsequently optimizing their performance by determining their optimal parameters using the GridSearch cross-validation method. It is better to check the performance of all models by finding their optimal parameters, however, it is a very time-consuming process.

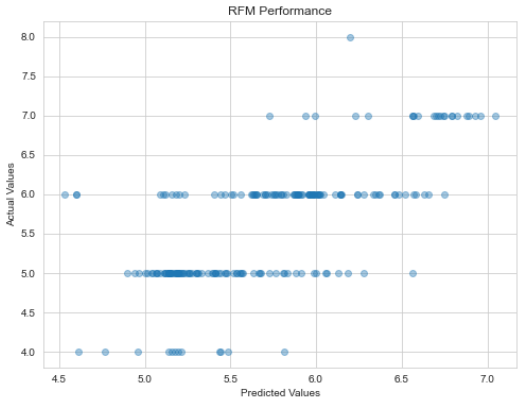
```python
# Random Forest Parameters
rf_param = {
    'n_estimators': [25, 50, 75, 100, 125, 150, 175, 200, 250, 275, 300, 350],
    'max_depth': [1, 5, 10, None],
    'min_samples_split': [2, 5, 10, 15],
    'min_samples_leaf': [1, 2, 4, 5],
    'max_features': ['auto', 'sqrt', 'log2']
}

# Gradient Boosting Parameters
gb_param = {
    'n_estimators': [25, 50, 75, 100, 125, 150, 175, 200, 250, 275, 300, 350],
    'max_depth': [1, 3, 5, 10, None],
    'min_samples_split': [2, 5, 7, 10],
    'min_samples_leaf': [1, 2, 4],
    'learning_rate': [0.25, 0.1, 0.05, 0.01],
    'max_features': ['auto', 'sqrt', 'log2']
}
```

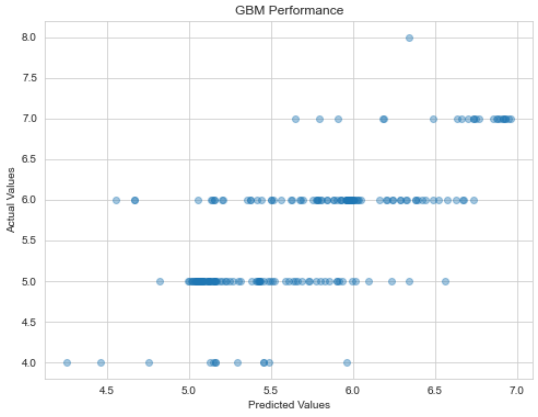
To optimize the performance of the Random Forest (RF) and Gradient Boosting (GB) models, the GridSearchCV method will be utilized to determine the best set of parameters for each model. Specifically, the algorithm will evaluate a total of 2,304 different models for the RF model and 8,640 different models for the GB model.

```python
rf = RandomForestRegressor()
gb = GradientBoostingRegressor()

# Finding the best parameters:
rf_grid = GridSearchCV(rf, rf_param, cv=5, n_jobs=-1)
rf_grid.fit(X_train, y_train)

gb_grid = GridSearchCV(gb, gb_param, cv=5, n_jobs=-1)
gb_grid.fit(X_train, y_train)

print('Random Forest - Best Parameters: ', rf_grid.best_params_)
print('Gradient Boosting - Best Parameters: ', gb_grid.best_params_)
```

```
Random Forest - Best Parameters:  {'max_depth': None, 'max_features': 'log2',
'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}
Gradient Boosting - Best Parameters:  {'learning_rate': 0.01, 'max_depth': 1
0, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_
estimators': 350}
CPU times: total: 1min 54s
Wall time: 40min 42s
```

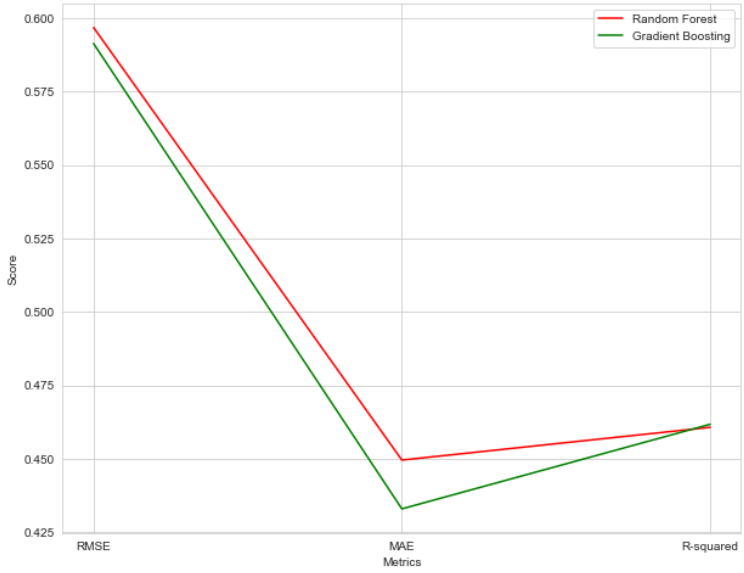
After setting the optimal parameters for the selected models, it's time to evaluate their performance. Here are scatter plots for both models:





It is important to note that quality values are discrete integers, while model predictions return continuous float numbers. As a result, predicted values may spread across multiple actual quality values. The models perform well in detecting wine qualities of 5, 6, and 7, but may have difficulty detecting quality 4 wines, resulting in errors in the predictions.

Here is the graph representing performance of the models on the test set:

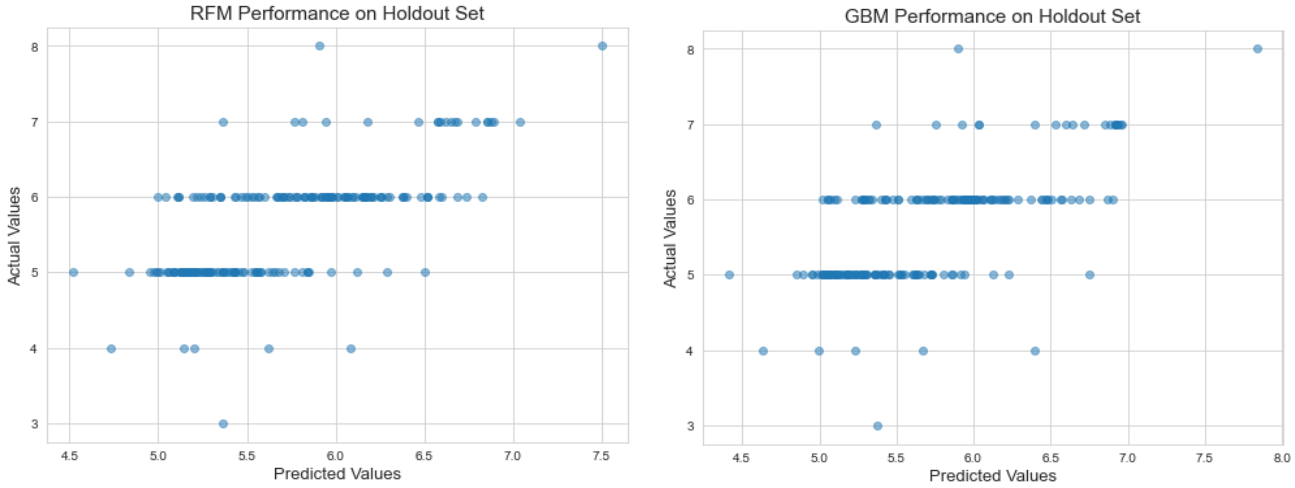


The graph above clearly displays the performance comparison of Gradient Boosting and Random Forest models, where Gradient Boosting model has outperformed the Random Forest model in all evaluation metrics. This result suggests that the Gradient Boosting model is a better choice over the Random Forest model for the given dataset. However, it is important to note that the difference in the metrics is relatively small, indicating that both models perform similarly. Further analysis and evaluation are necessary to determine the practical significance of this performance difference.

5. Comparison of models

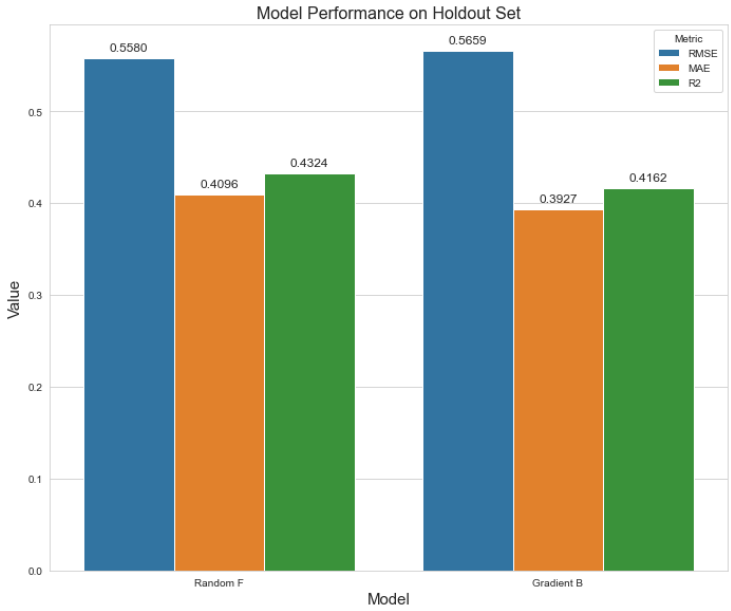
The primary objective of identifying the best models is to select the model that exhibits the best performance when applied to real-time data. It is essential to ensure that the chosen model generalizes well and does not overfit the training data. To avoid overfitting, it is necessary to test the models on a holdout set, and further refine them by performing hyperparameter tuning. Once the best model is selected, it is crucial to ensure that the model remains up-to-date and continuously performs well by periodically retraining it with the latest data.

Here are scatterplots of the models:



The results are similar to the previous results which suggests that models have good generalization ability.

Here are the evaluation metrics:



This graph presents different results compared to the previous one, indicating that the Random Forest model performs slightly better on the live data than Gradient Boosting model. This suggests that the Random Forest model has a more robust ability to avoid overfitting.

6. Conclusion

In this study, seven different models were evaluated by applying cross-validation and comparing their performance metrics including root mean squared error (RMSE), mean absolute error (MAE), and R-squared (R^2). Among the evaluated models, Random Forest and Gradient Boosting were identified as the best performing models based on their performance metrics. The hyperparameters of the selected models were then optimized to maximize their predictive power. The optimized models were then applied to predict the holdout set. The evaluation of the predictive performance of the optimized models on the holdout set revealed that Random Forest performed slightly better than Gradient Boosting. Therefore, based on the findings of this study, it can be concluded that RF model is a more suitable option for predicting the target variable in the given dataset. Although the current approach provides useful insights into the performance of the evaluated models, it is important to note that there may be other approaches that could potentially yield even more accurate results. For instance, optimizing hyperparameters for a broader range of models or trying different feature engineering techniques could potentially improve the predictive power of the models.

Used Sources

Cortez,Paulo, Cerdeira,A., Almeida,F., Matos,T. & Reis,J. (2009). Wine Quality. UCI Machine Learning Repository. https://doi.org/10.24432/C56S3T

Gábor Békés and Gábor Kézdi. (2021). *Data Analysis for Business, Economics and Policy*

Dhaliwal, P., Sharma, S., & Chauhan, L. (2022). *Detailed Study of Wine Dataset and its Optimization*