Evaluation of Methods Used to Model Wine Quality

Introduction

This project aims to use random forest regressors and support vector regressors to accurately predict wine quality and its most important features, examine differences between models, and investigate unbalanced data.

Wine quality monitoring is important to determine pricing, ensure safety, and regulatory compliance. Wine quality changes are particularly important, given climate change will affect key factors (temperature, precipitation, disease) that control grape quality and yield. Further, given that wine is culturally, historically, and economically significant to many wine-growing regions, a decrease in wine quality may drastically impact the region. The wine sector in Portugal is a major employer, with approximately 150,000 (15% of people employed in agriculture) employees involved, globally Portugal is one of the top 10 wine exporters (Guo & Zhong, 2024). Therefore, monitoring wine quality, and the dominant features predicting quality is essential.

Data

This project utilises two separate datasets related to red and white wine samples from the UC Irvine Machine learning database (Cortez, Cerdeira, Almeida, Matos, & Reis, 2009). These wine samples are vinho verde (‘green wine) from the northwestern Minho region of Portugal. The Vinho Verde Viticulture Commission (CVRVV) oversees regulation and control of wine quality from this region. The datasets used by this model were collected from May 2004 to February 2007 using samples tested by CVRVV. Each sample was evaluated by a minimum of three testers, with a score given from 0-10 with 0 being ‘very bad’ and 10 being ‘excellent’, this data was recorded by a computerised system and exported onto a csv (Cortez et al., 2009).

Before the modelling stage, data frames (Table 1), data descriptions, and histograms (Figure 1) were printed to aid in choosing the most suitable model, and to determine whether reprocessing is required. The red and white wine datasets contain the target variable, quality, and 10 continuous features: fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH and sulphates. There are no missing values in these datasets. Given that wine quality is given as an integer between 0-10, this data will lend itself to regression models. Both datasets have the same features. Therefore, it is useful to combine the datasets, providing more data to the models and increasing the number of extreme values. When combining the datasets, another feature ‘is_red’ is added, with red wine instances given a score of 1 (code snippet 1), and white wine instances given a score of 0. This enables investigation into whether wine colour affects quality.

```
df_red_wine = pd.read_csv(csv_path, sep = ';')
df_red_wine['is_red'] = np.ones(len(df_red_wine)) #creates new feature 'is_red', filled with ones (for red wine)
display(df_red_wine['quality'].describe())
display(df_red_wine)
```

Code snippet 1. Preparing red wine data for combination

A

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	is_red
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	1.0
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5	1.0
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5	1.0
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6	1.0
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5	1.0
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6	1.0
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6	1.0
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5	1.0
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6	1.0

B

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	is_red
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.00100	3.00	0.45	8.8	6	0.0
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.99400	3.30	0.49	9.5	6	0.0
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.99510	3.26	0.44	10.1	6	0.0
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	9.9	6	0.0
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	9.9	6	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4893	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	6	0.0
4894	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	5	0.0
4895	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	6	0.0
4896	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	7	0.0
4897	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	6	0.0

Table 1A and 1B. printed data frames for red (A) and white (B) wine data, both have the same features meaning they can be combined with ease

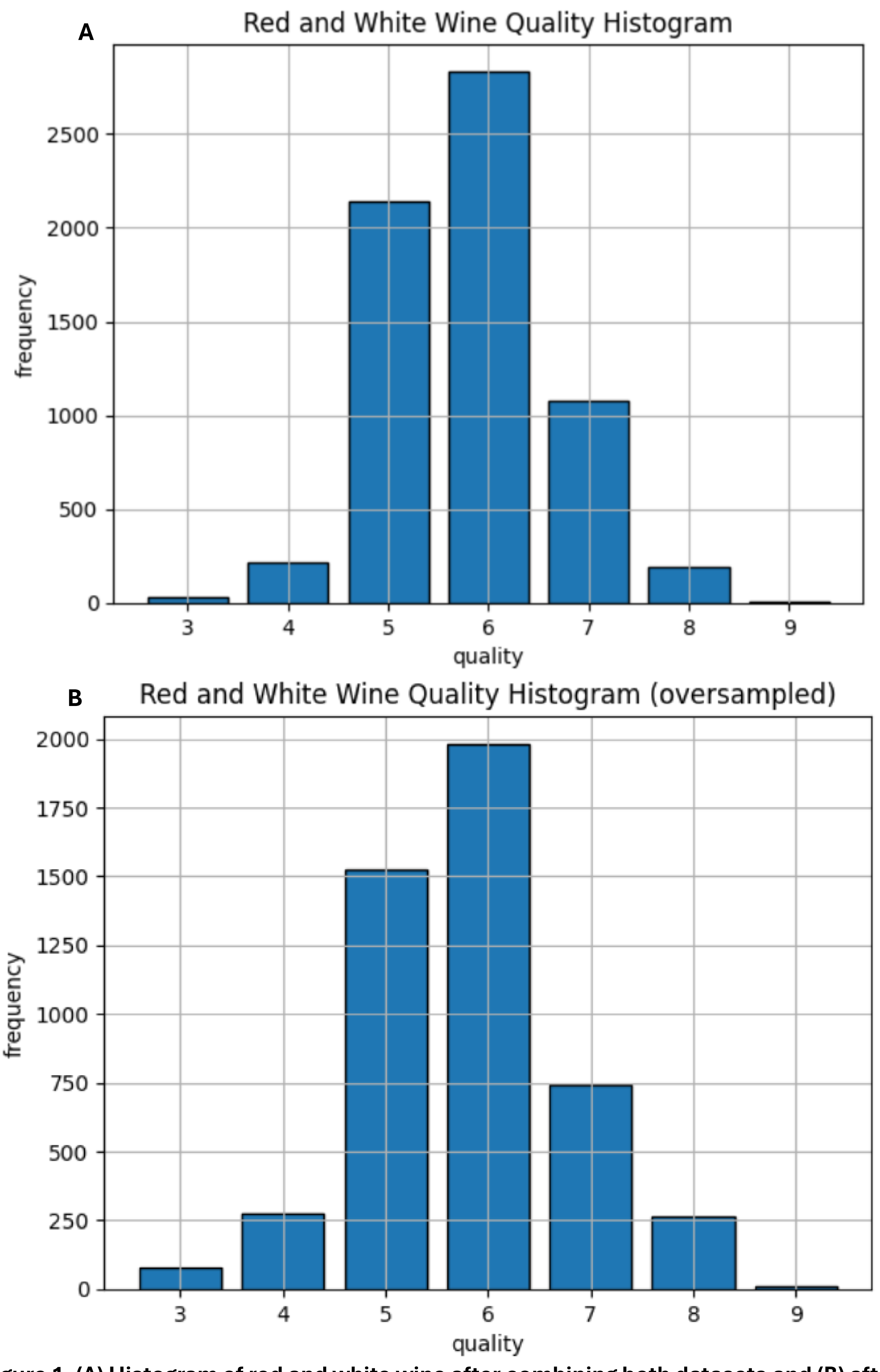
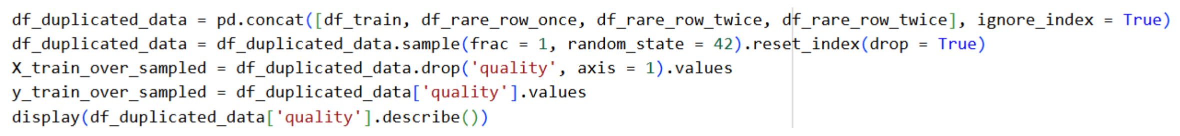


Figure 1. (A) Histogram of red and white wine after combining both datasets and (B) after oversampling. The unbalanced nature of the dataset is evident, with very little data for quality ratings of 3 and 9, which are increased in B.

To increase the representation of the underpredicted values, 'rare' quality values, were doubled (qualities 4, and 8), and quadrupled (qualities 3 and 9) (Figure 1B), to produce a new dataset ('df_duplicated_data') so that the model 'sees' these rare values more often (oversampling) (code snippet 2). This was done before splitting the data into train and test data to ensure duplicated values are only present in the training data, preventing data leakage. This data frame will be analysed separately to evaluate the effect of training models on more balanced data.



```
df_duplicated_data = pd.concat([df_train, df_rare_row_once, df_rare_row_twice, df_rare_row_twice], ignore_index = True)
df_duplicated_data = df_duplicated_data.sample(frac = 1, random_state = 42).reset_index(drop = True)
X_train_over_sampled = df_duplicated_data.drop('quality', axis = 1).values
y_train_over_sampled = df_duplicated_data['quality'].values
display(df_duplicated_data['quality'].describe())
```

Code snippet 2. Section of code used to generate oversampled data, random_state = 42 ensures duplicated members are not placed adjacently.

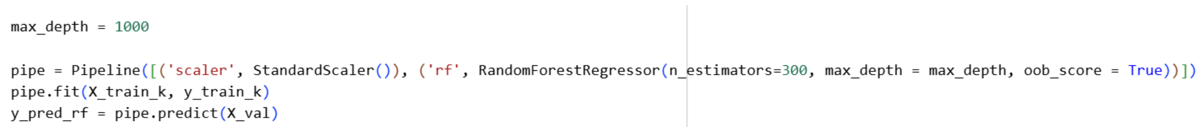
Data was scaled within the model pipelines ensuring features have consistent ranges and ensures fair contributions from all features.

Modelling

Two models, random forest regressor and support vector regression were used for this project. For modelling, 30% of the data was used in testing the model, and 70% was used for training, striking the balance between size of the training data and potential overfitting (Gholamy, Kreinovich, & Kosheleva, 2018). Scikit-learn pipelines were also used to scale the data set, and the pipelines were then instantiated and trained on the data (code snippet 2).

Random forest Regressor is a supervised ensemble learning method, built from a collection of many decision trees. Each tree is trained on a bootstrap sample, and a random subset of features is used at each split. This increases model diversity and reduces overfitting. Hence, random forest regressors are robust to noise, capable at handling complex relationships, and generally have high classification accuracy (Liu, Wang, & Zhang, 2012). A benefit of using random forests is the ease of determining feature importances, which were computed for the 'best_model.'

As this model trained on imbalanced data, I modified the following hyperparameters (Code snippet 3): oob_score, max_depth, and n_estimators, on a trial-and-error approach, choosing the best combination by observing the change in error metrics as I changed these parameters. I found that increasing the max_depth to 1000 (to capture more detail) and increasing n_estimators to 300 (averages over more trees, reducing error) helped reduce the root mean squared error (RMSE) by approximately 0.05, beyond which, any changes were minimal (<0.01).



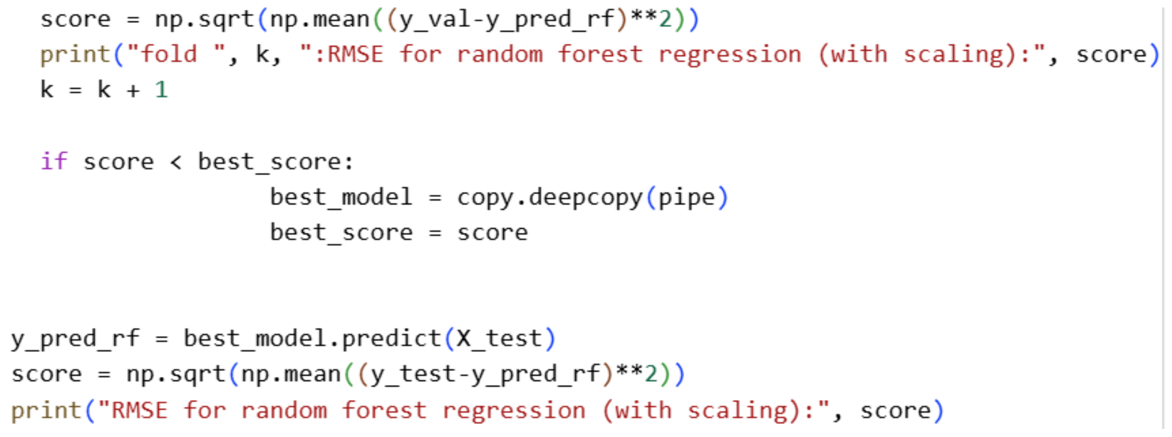
```
max_depth = 1000

pipe = Pipeline([('scaler', StandardScaler()), ('rf', RandomForestRegressor(n_estimators=300, max_depth = max_depth, oob_score = True))])
pipe.fit(X_train_k, y_train_k)
y_pred_rf = pipe.predict(X_val)
```

Code snippet 3. Demonstration of pipelines and hyperparameter tuning

Support vector regression (SVR) is a supervised learning method that predicts continuous target values by fitting a function within a specific tolerance. Support vectors define a margin around the predictions, making them robust to noise. SVR is well-suited to small to medium datasets (up to 10,000 samples) making it an ideal choice for the wine data.

To help fine-tune this model, I decided to modify C, where increasing C makes the model fit data more closely (risks overfitting), whereas decreasing C increases regularisation (default = 1) to help reduce underfitting. However little change was observed in the error metrics upon this modification and therefore the default parameters were used.

To reduce overfitting, I used cross validation, which repeatedly trains and tests the model on different subsets of the data (code snippet 4), thus enabling selection of the 'best_model' with the lowest RMSE Several error metrics: RMSE, R^2, mean absolute error and regression were then calculated enabling evaluation of the model performance. Regression slopes for each model were plotted with translucent data points to enhance visual clarity (Figure 2A, 2B). These slopes were compared to a line where predictions made by the model correspond exactly to the actual y values to visualise how closely each model follows the ideal perfect prediction line.



```
    score = np.sqrt(np.mean((y_val-y_pred_rf)**2))
    print("fold ", k, ":RMSE for random forest regression (with scaling):", score)
    k = k + 1

    if score < best_score:
                    best_model = copy.deepcopy(pipe)
                    best_score = score


  y_pred_rf = best_model.predict(X_test)
  score = np.sqrt(np.mean((y_test-y_pred_rf)**2))
  print("RMSE for random forest regression (with scaling):", score)
```

Code snippet 4. Section of cross validation code for the random forest model

Next, REC curves were plotted for each model (Figure 2C), showing for a given error, the percentage of correct predictions. Following this, feature ranking was conducted to determine feature importances (Figure 2D). The exact same steps were conducted on the oversampled data.

Results

The following results were obtained from the error metrics. For the random forest, the RMSE and mean absolute errors were 0.62 and 0.45 respectively, showing that, on average the typical size of prediction errors is 0.62 or 0.45 depending on the chosen error metric. The R^2 metric was 0.49 showing that 49% of the variance in wine quality is explained by the model. For the SVR, the RMSE and mean absolute errors were 0.69 and 0.52 respectively, The R^2 metric was 0.38 showing that 38% of the variance in wine quality is explained by the model.

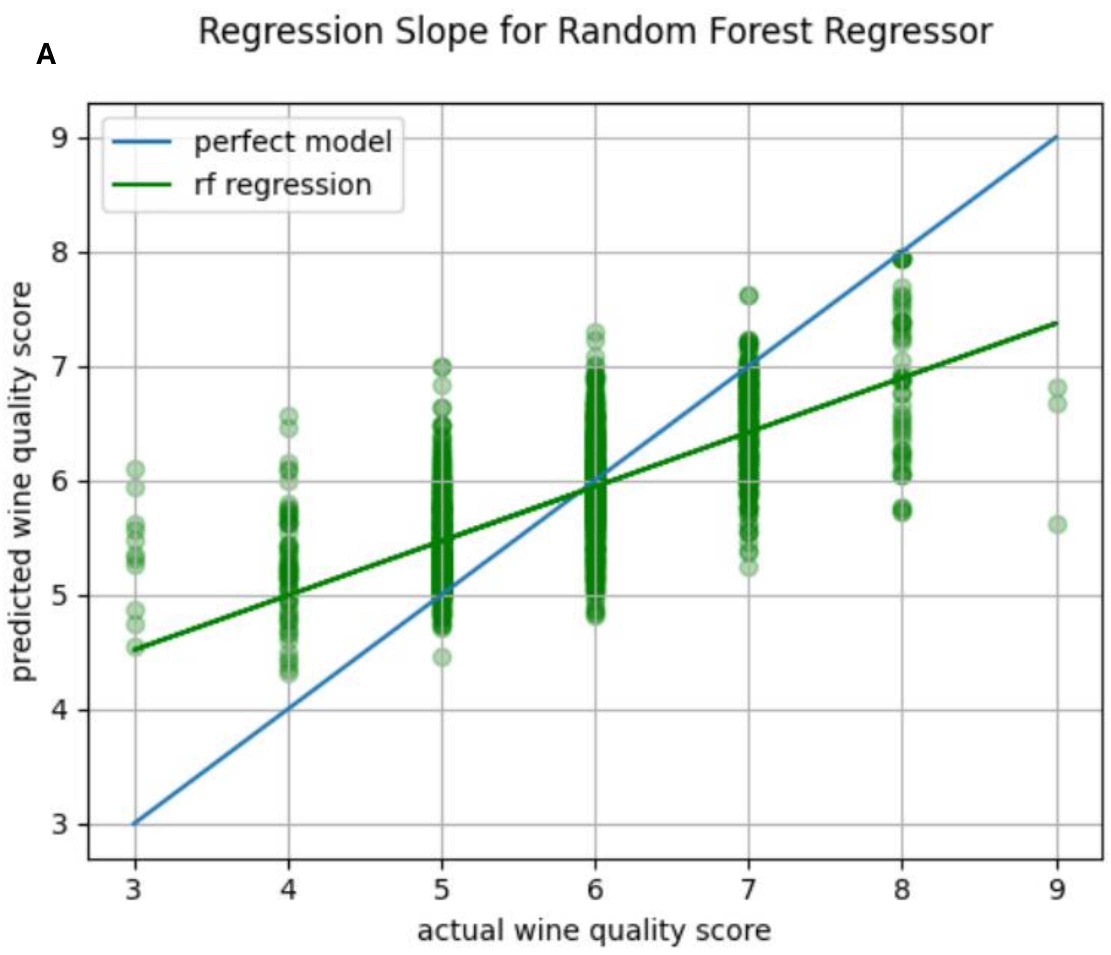
For both regressions, the slope differs significantly from the 'perfect prediction.' This is particularly significant for the lowest and highest qualities, where low qualities are overpredicted and high qualities are underpredicted (Figure 2A, 2B).

The REC curves for both SVR and random forest regressor are very similar (Figure 2C, with the random forest regressor having a slightly higher prediction accuracy. At an error tolerance of 1, >85% of the quality is predicted accurately, and at a tolerance of 2, almost 100% of the data is predicted within this error.

The most important feature by a significant margin is alcohol, followed by volatile acidity, it is evident that wine colour has very little effect on wine quality (Figure 2D)

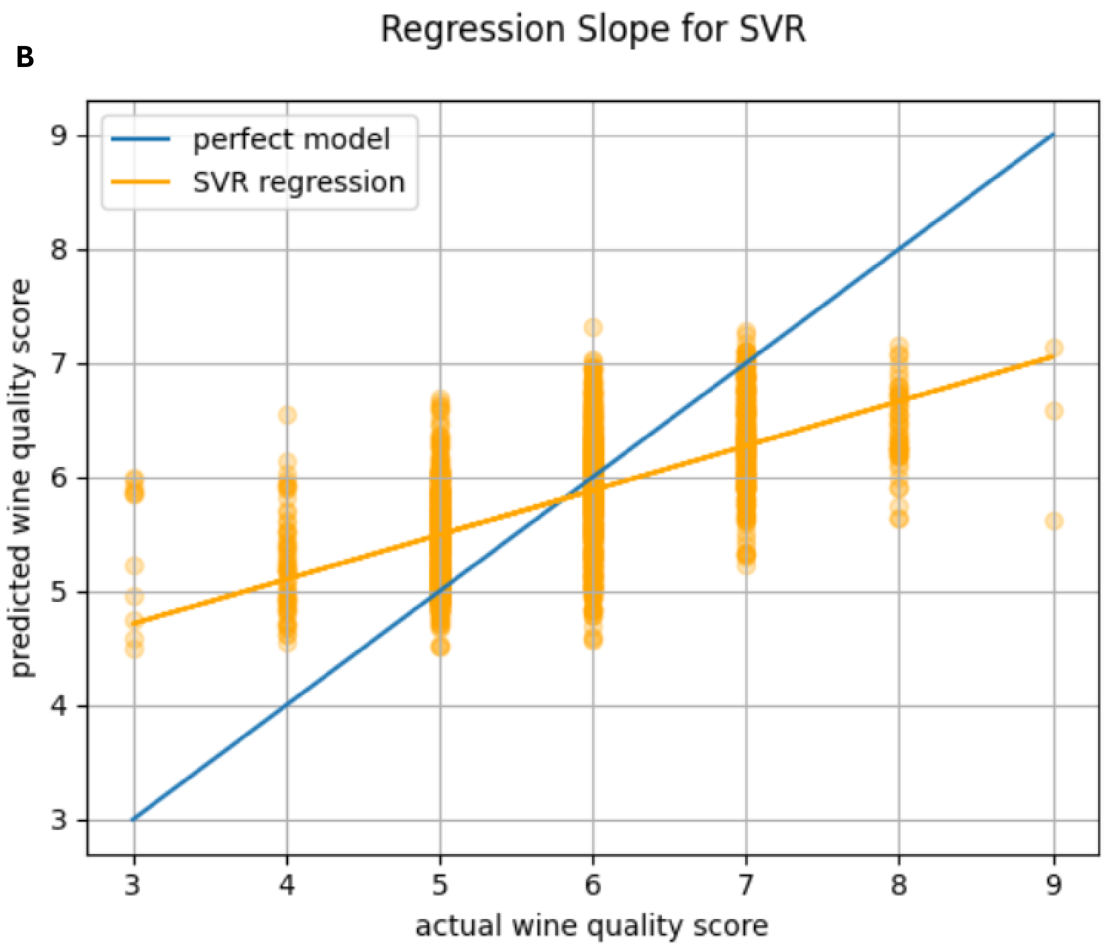
Very similar results were obtained from the error metrics for the model trained on oversampled data. For the random forest and SVR, the error metrics were identical to two significant figures. Both regression curves were also almost identical to those from the models that were not trained on the oversampled data.

A



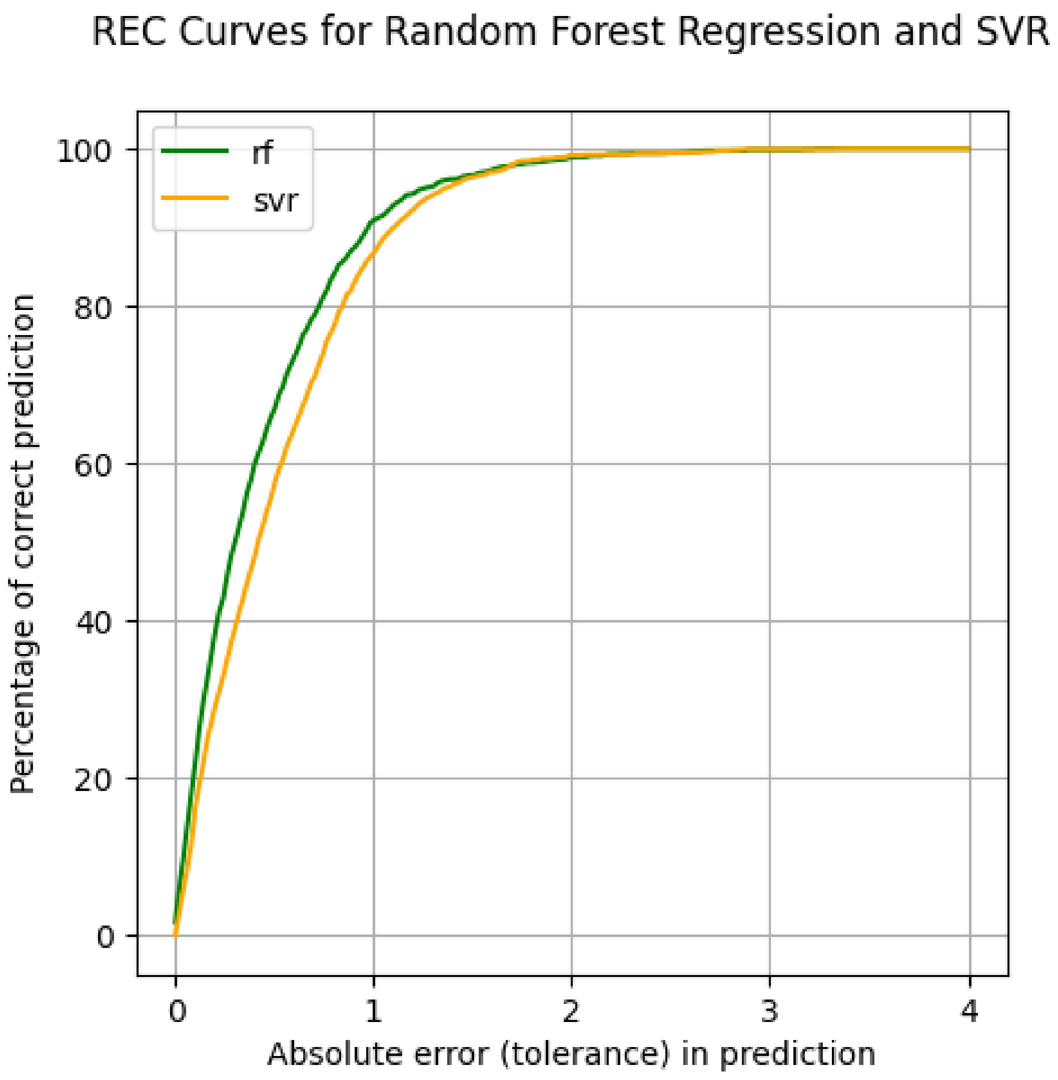
Regression Slope for Random Forest Regressor
perfect model
rf regression
predicted wine quality score
actual wine quality score
3
4
5
6
7
8
9

B



Regression Slope for SVR
perfect model
SVR regression
predicted wine quality score
actual wine quality score
3
4
5
6
7
8
9

C


REC Curves for Random Forest Regression and SVR
rf
svr
Percentage of correct prediction
Absolute error (tolerance) in prediction
0
20
40
60
80
100
0
1
2
3
4

D

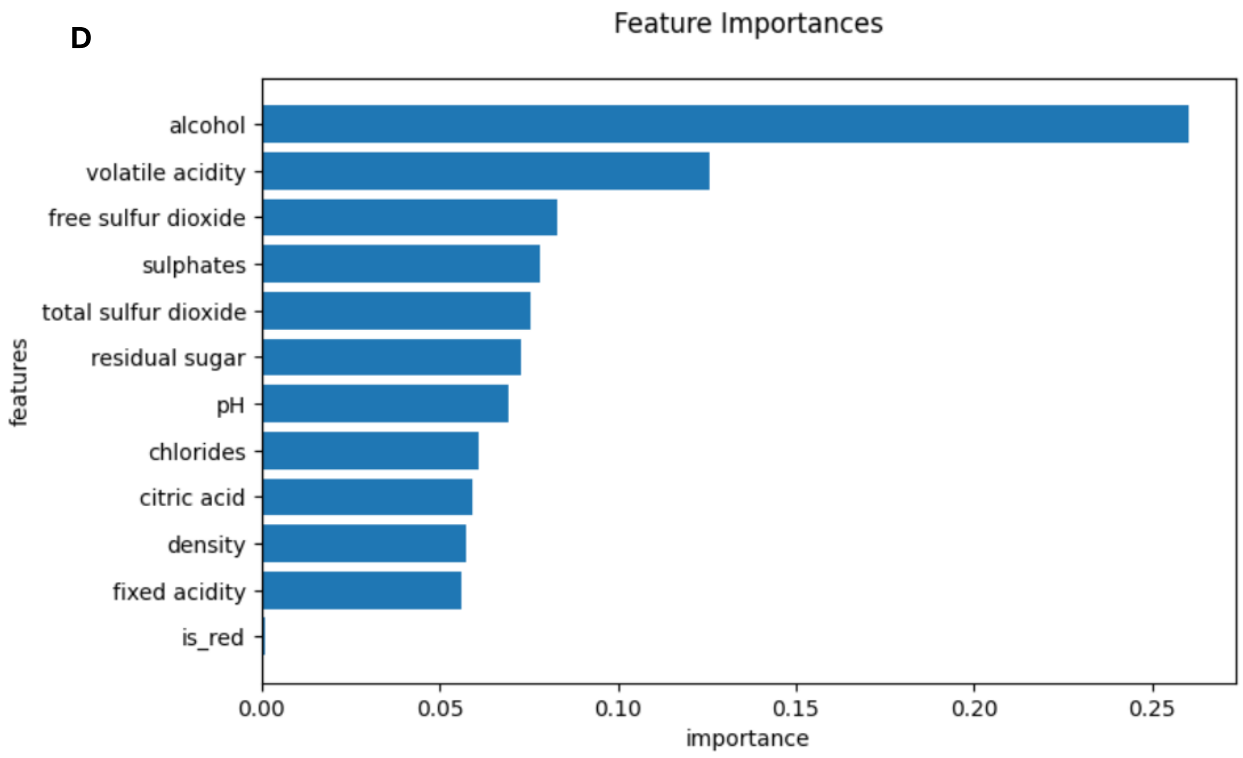

Feature Importances
alcohol
volatile acidity
free sulfur dioxide
sulphates
total sulfur dioxide
residual sugar
pH
chlorides
citric acid
density
fixed acidity
is_red
features
importance
0.00
0.05
0.10
0.15
0.20
0.25

Figure 2. (A) Regression slope (green) for random forest regressor showing divergence from the perfect model (blue), (B) Regression slope for SVR (orange), slightly shallower gradient than model in (A). (C) REC curves plotted for random forest regression (green) and SVR (orange), (D) Feature importances for wine quality predictions

Discussion

In the discussion, RMSE scores will be used to determine prediction accuracy as it is the preferred metric for normally distributed data (Hodson, 2022).

From the results, it is evident that the random forest regressor is better suited to this data, producing better RMSE (improvement of 0.07), and R^2 (11% improvement) scores. Therefore, the random forest regressor predicts quality more accurately and also explains more of the data spread. Nevertheless, in this context, both RMSE score's can said to be significant, given that quality rankings are based on taste, which is complex and varies from person to person (Smith, 2007). Furthermore, although the R^2 values appear low (0.49 and 0.38), values quoted as 'acceptable' in the social science and financial studies can range from as low as 10%-30% , or 40%-70% respectively (Gupta, Stead, & Ganti, 2024). Therefore, our R^2 values for random forest and SVR show that these models capture a reasonable amount of variability within the data given the human influence on the study.

Examining the linear regression curves for both models (Figure 2A, 2B) proves that significant error occurs at the margins of the data, rare quality values are overpredicted (for low qualities) and underpredicted (for higher qualities). This is reflected in the relatively high RMSE values, which punish outliers heavily, compared to the mean absolute error which does not. Generally, qualities in the middle (5,6,7) are well predicted, as shown by how closely the regression line follows the 'perfect prediction.'

The REC curves build on the evidence provided by the error metrics in determining which model is more effective (Figure 2C). At almost every tolerance, the percentage of correct predictions for the random forest model is greater, although both show similar curves. This, along with similarities in the regression lines suggest that they are both affected by the same factors: imbalance in data, and the natural unpredictability caused by individual taste.

Feature ranking (Figure 2D) suggests that the three most significant factors affecting wine quality: alcohol, volatile acidity, and free sulfur dioxide consistently appear as some of the most significant factors in the literature (Cortez et al., 2009). However, it may also be the case that different features affect wine quality differently depending on wine colour, as suggested by (Cortez et al., 2009). Therefore, significant caution must be given when examining these results.

In this model, an attempt was also made to build the best possible model by modifying the hyperparameters which was successful, however, the oversampling technique did little to improve error. All error metrics and graphs were almost the same as for the models that were not trained on the oversampled data, likely due to the fact that these data points were still incredibly rare and because the test distribution remained unchanged, Therefore, it will not be discussed further.

Conclusion

From this project, it can be concluded that the random forest regressor is more effective at predicting wine quality than SVR for this dataset. This report also emphasises the importance of hyperparameter tuning and effective balancing of data to improve model accuracy when presented with imbalanced data influenced by human sensory evaluations, introducing subjectivity into target values. My method of oversampling by duplicating values in the training dataset was relatively ineffective, likely due to the datapoints still remaining rare. Thus, further investigation into this method, or more advanced resampling would prove beneficial, although one must exercise caution in modifying the training data beyond what is acceptable. Feature ranking has been proven to be a highly useful tool that can be implemented following a random forest regression, aiding those involved in the wine industry to focus on qualities contributing most significantly to wine quality.

References

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems, 47*(4), 547-553. doi:https://doi.org/10.1016/j.dss.2009.05.016

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

Gholamy, A., Kreinovich, V., & Kosheleva, O. (2018). Why 70/30 or 80/20 relation between training and testing sets: A pedagogical explanation.

Guo, G., & Zhong, L. (2024). Analysis of the Profit Model of the Portuguese Wine Business. *Information Systems and Economics, 5*(2), 80-85.

Gupta, A., Stead, T. S., & Ganti, L. (2024). Determining a meaningful R-squared value in clinical medicine. *Academic Medicine & Surgery*.

Hodson, T. O. (2022). Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not. *Geosci. Model Dev., 15*(14), 5481-5487. doi:10.5194/gmd-15-5481-2022

Liu, Y., Wang, Y., & Zhang, J. (2012). *New Machine Learning Algorithm: Random Forest*, Berlin, Heidelberg.

Smith, B. C. (2007). The objectivity of tastes and tasting. *Questions of taste: The philosophy of wine*, 41-77.